ЛЕКЦИИ ПО МАТЕМАТИЧЕСКОЙ ЛОГИКЕ
И ТЕОРИИ АЛГОРИТМОВ

Н. К. Верещагин, А. Шень

ВЫЧИСЛИМЫЕ ФУНКЦИИ

Москва, 1999
МЦНМО

ББК 22.12

В31 **Н. К. Верещагин, А. Шень.** Лекции по математической логике и теории алгоритмов. Часть 3. Вычислимые функции. М.: МЦНМО, 1999, **176** с.

Книга написана по материалам лекций и семинаров, проводившихся авторами для студентов младших курсов мехмата МГУ. В ней рассказывается об основных понятиях общей теории вычислимых функций (вычислимость, разрешимость, перечислимость, универсальные функции, нумерации и их свойства, m-полнота, теорема о неподвижной точке, арифметическая иерархия, вычисления с оракулом, степени неразрешимости) и о конкретных вычислительных моделях (машины Тьюринга, рекурсивные функции). Изложение рассчитано на учеников математических школ, студентов-математиков и всех интересующихся основами теории алгоритмов. Книга включает себя около 90 задач различной трудности.

Тексты, составляющие книгу, являются свободно распространяемыми и доступны по адресу

`ftp://ftp.mccme.ru/users/shen/logic/comput`

ISBN 5-900916-39-1

Книга издана при финансовой поддержке
Российского фонда фундаментальных исследований
(проект 98-01-14119)

Издательство Московского Центра
непрерывного математического образования

Технический редактор В. В. Шувалов

Лицензия ЛР №071150 от 11.04.95 г.
Подписано в печать ??.??.??. Формат 84 × 108/32.
Печать офсетная. Печ. л. 5,5
Тираж ????. Заказ №?.

МЦНМО
121002, Москва, Большой Власьевский пер., 11

Оглавление

Предисловие 6

1. Вычислимость, разрешимость и перечислимость 8
   - 1.1. Вычислимые функции 8
   - 1.2. Разрешимые множества 9
   - 1.3. Перечислимые множества 10
   - 1.4. Перечислимые и разрешимые множества . 13
   - 1.5. Перечислимость и вычислимость 14

2. Универсальные функции и неразрешимость 17
   - 2.1. Универсальные функции 17
   - 2.2. Диагональная конструкция 18
   - 2.3. Перечислимое неразрешимое множество . . 20
   - 2.4. Перечислимые неотделимые множества . . 21
   - 2.5. Простые множества: конструкция Поста . 22

3. Нумерации и операции 24
   - 3.1. Главные универсальные функции 24
   - 3.2. Вычислимые последовательности функций 28
   - 3.3. Главные универсальные множества 29

4. Свойства главных нумераций 32
   - 4.1. Множества номеров 32
   - 4.2. Новые номера старых функций 36
   - 4.3. Изоморфизм главных нумераций 39
   - 4.4. Перечислимые свойства функций 41

5. Теорема о неподвижной точке 45
   - 5.1. Неподвижная точка и отношения эквивалентности 45
   - 5.2. Программа, печатающая свой текст 47
   - 5.3. Системный трюк: ещё одно доказательство 50
   - 5.4. Несколько замечаний 53

6. m-сводимость и свойства перечислимых множеств 59
6.1. m-сводимость 59
6.2. m-полные множества 60
6.3. m-полнота и эффективная неперечислимость 62
6.4. Изоморфизм m-полных множеств 66
6.5. Продуктивные множества 68
6.6. Пары неотделимых множеств 72

7. Вычисления с оракулом 76
7.1. Машины с оракулом 76
7.2. Эквивалентное описание 79
7.3. Релятивизация 81
7.4. $\mathbf{0}'$-вычисления 85
7.5. Несравнимые множества 88
7.6. Теорема Мучника – Фридберга: схема конструкции 92
7.7. Теорема Мучника – Фридберга: выигрышные условия 94
7.8. Теорема Мучника – Фридберга: метод приоритета 96

8. Арифметическая иерархия 98
8.1. Классы Σ_n и Π_n 98
8.2. Универсальные множества в Σ_n и Π_n . . . 101
8.3. Операция скачка 102
8.4. Классификация множеств в иерархии . . . 108

9. Машины Тьюринга 112
9.1. Зачем нужны простые вычислительные модели? 112
9.2. Машины Тьюринга: определение 113
9.3. Машины Тьюринга: обсуждение 115
9.4. Ассоциативные исчисления 118
9.5. Моделирование машин Тьюринга 120
9.6. Двусторонние исчисления 124
9.7. Полугруппы, образующие и соотношения . 125

10. Арифметичность вычислимых функций 129
10.1. Программы с конечным числом переменных 129
10.2. Машины Тьюринга и программы 132
10.3. Арифметичность вычислимых функций . . 134
10.4. Теоремы Тарского и Гёделя 138
10.5. Прямое доказательство теорем Тарского и Гёделя 140
10.6. Арифметическая иерархия и перемены кванторов 142

11. Рекурсивные функции 145
11.1. Примитивно рекурсивные функции 145
11.2. Примеры примитивно рекурсивных функций 146
11.3. Примитивно рекурсивные множества . . . 147
11.4. Другие виды рекурсии 150
11.5. Машины Тьюринга и рекурсивные функции 153
11.6. Частично рекурсивные функции 155
11.7. Вычислимость с оракулом 159
11.8. Оценки скорости роста. Функция Аккермана 162

Литература 166

Предметный указатель 168

Указатель имён 174

Предисловие

Предлагаемая вашему вниманию книга написана по материалам лекций для младшекурсников, которые читались авторами в разные годы на механико-математическом факультете МГУ. (В этой серии уже вышла книга «Начала теории множеств»; готовится к печати книга «Языки и исчисления».)

Теория вычислимых (с помощью компьютеров) функций появилась в 1930-е годы, когда никаких компьютеров ещё не было. Первые компьютеры были разработаны в 1940-х годах, и среди их разработчиков был английский математик Алан Тьюринг, один из создателей теории вычислимых функций. В 1936 году он описал абстрактные машины, которые теперь называют *машинами Тьюринга*, и можно полагать, что идея хранимой в памяти компьютера программы была подсказана доказательством теоремы о существовании универсальной машины Тьюринга.

Уже поэтому основные понятия теории вычислимости (или, как говорят, общей теории алгоритмов) достойны внимания математиков и программистов. Но эта теория имеет и более широкий культурный аспект. Один из её основателей, американский математик Эмиль Пост, писал в 1944 году, что «формулировка этого понятия [вычислимости] может сыграть в истории дискретной математики роль, уступающую по значению лишь формулировке понятия натурального числа».

Пожалуй, сейчас это высказывание Поста выглядит преувеличением: за последние десятилетия стало ясно, что различие между быстро и долго решаемыми задачами не менее философски важно, чем различие между алгоритмически разрешимыми и неразрешимыми, и теория сложности вычислений стала одной из центральных в логике (и вообще в математике).

Мы сознательно не касаемся теории сложности вычислений — это большая и отдельная тема. Вместо этого мы попытались отобрать центральные понятия и факты общей теории алгоритмов и изложить их понятно, стараясь не заслонять простые общие идеи техническими деталями. Мы не предполагаем никаких специальных предварительных знаний, хотя рассчитываем на некоторый уровень математической культуры (и не объясняем, скажем, что такое множество, функция или действительное число).

Надеемся, что первое знакомство с теорией алгоритмов доставит удовольствие. Для тех, кто захочет продолжить знакомство с этой теорией (являющейся центральной частью математической логики), мы указываем на с. 166 некоторые изданные на русском языке книги.

Авторы пользуются случаем поблагодарить своего учителя, Владимира Андреевича Успенского, лекции, тексты и высказывания которого повлияли на них (и на содержание этой книги), вероятно, даже в большей степени, чем авторы это осознают.

При подготовке текста использованы записи А. Евфимьевского и А. Ромащенко (который также прочёл предварительный вариант книги и нашёл там немало ошибок).

Оригинал-макет книги подготовлен её редактором, В. В. Шуваловым; без его настойчивости (вплоть до готовности разделить ответственность за ошибки) оригинал-макет вряд ли появился бы к какому-либо сроку.

Авторы признательны École Normale Supérieure de Lyon (Франция) за поддержку и гостеприимство во время написания этой книги.

Издание книги стало возможным благодаря Российскому фонду фундаментальных исследований, а также И. В. Ященко, который уговорил авторов подать туда заявку.

Наконец, мы благодарим сотрудников, аспирантов и студентов кафедры математической логики мехмата МГУ, а также всех участников наших лекций и семинаров и читателей предварительных вариантов этой книги.

Просим сообщать о всех ошибках и опечатках авторам (электронные адреса `ver@mccme.ru`, `shen@mccme.ru`; почтовый адрес: Москва, 121002, Большой Власьевский пер., 11, Московский центр непрерывного математического образования).

Н. К. Верещагин, А. Шень

1. Вычислимые функции, разрешимые и перечислимые множества

1.1. Вычислимые функции

Функция f с натуральными аргументами и значениями называется *вычислимой*, если существует алгоритм, её вычисляющий, то есть такой алгоритм A, что

- если $f(n)$ определено для некоторого натурального n, то алгоритм A останавливается на входе n и печатает $f(n)$;
- если $f(n)$ не определено, то алгоритм A не останавливается на входе n.

Несколько замечаний по поводу этого определения:

1. Понятие вычислимости определяется здесь для частичных функций (областью определения которых является некоторое подмножество натурального ряда). Например, нигде не определённая функция вычислима (в качестве A надо взять программу, которая всегда зацикливается).

2. Можно было бы изменить определение, сказав так: «если $f(n)$ не определено, то либо алгоритм A не останавливается, либо останавливается, но ничего не печатает». На самом деле от этого ничего бы не изменилось (вместо того, чтобы останавливаться, ничего не напечатав, алгоритм может зацикливаться).

3. Входами и выходами алгоритмов могут быть не только натуральные числа, но и двоичные строки (слова в алфавите $\{0, 1\}$), пары натуральных чисел, конечные последовательности слов и вообще любые, как говорят, «конструктивные объекты». Поэтому аналогичным образом можно определить понятие, скажем, вычислимой функции с двумя натуральными аргументами, значениями которой являются рациональные числа.

Для функций, скажем, с действительными аргументами и значениями понятие вычислимости требует специального определения. Здесь ситуация сложнее, определения могут быть разными, и мы о вычислимости таких функций говорить не будем. Отметим только, что, например, синус (при разумном определении вычислимости) вычислим, а функция $\operatorname{sign}(x)$, равная -1, 0 и 1 при $x < 0$, $x = 0$ и $x > 0$ соответственно — нет. Точно так же требует специального определения вычислимость функций, аргументами которых являются бесконечные последовательности нулей и единиц и т. п.

4. Несколько десятилетий назад понятие алгоритма требовало специального разъяснения. Сейчас («компьютерная грамотность»?) такие объяснения всё равно никто читать не будет, поскольку и так ясно, что такое алгоритм. Но всё же надо соблюдать осторожность, чтобы не принять за алгоритм то, что им не является. Вот пример неверного рассуждения:

«Докажем», что всякая вычислимая функция f с натуральными аргументами и значениями может быть продолжена до всюду определённой вычислимой функции $g\colon \mathbb{N} \to \mathbb{N}$. В самом деле, если f вычисляется алгоритмом A, то следующий алгоритм B вычисляет функцию g, продолжающую f: «если A останавливается на n, то B даёт тот же результат, что и A; если A не останавливается на n, то B даёт результат (скажем) 0». (В чём ошибка в этом рассуждении?)

1.2. Разрешимые множества

Множество натуральных чисел X называется *разрешимым*, если существует алгоритм, который по любому натуральному n определяет, принадлежит ли оно множеству X.

Другими словами, X разрешимо, если его *характеристическая функция* $\chi(n) = (\textbf{if } n \in X \textbf{ then } 1 \textbf{ else } 0 \textbf{ fi})$ вычислима.

Очевидно, пересечение, объединение и разность разрешимых множеств разрешимы. Любое конечное множество разрешимо.

Аналогично определяют разрешимость множеств пар натуральных чисел, множеств рациональных чисел и т. п.

1. Докажите, что множество всех рациональных чисел, меньших числа e (основания натуральных логарифмов), разрешимо.

2. Докажите, что непустое множество натуральных чисел разрешимо тогда и только тогда, когда оно есть множество значений всюду определённой неубывающей вычислимой функции с натуральными аргументами и значениями.

Отметим тонкий момент: можно доказать разрешимость множества неконструктивно, не предъявляя алгоритма. Вот традиционный пример: множество тех n, для которых в числе π есть не менее n девяток подряд, разрешимо. В самом деле, это множество содержит либо все натуральные числа, либо все натуральные числа вплоть до некоторого. В обоих случаях оно разрешимо. Тем не менее мы так и не предъявили алгоритма, который по n узнавал бы, есть ли в π не менее n девяток подряд.

3. Использованы ли в этом рассуждении какие-то свойства числа π? Что изменится, если заменить слова «не менее n девяток» на «ровно n девяток (окружённых не-девятками)»?

Существуют ли неразрешимые множества? Существуют — просто потому, что алгоритмов (и поэтому разрешимых подмножеств натурального ряда) счётное число, а всех подмножеств натурального ряда несчётное число. Более конкретные примеры мы ещё построим.

1.3. Перечислимые множества

Множество натуральных чисел называется *перечислимым*, если оно перечисляется некоторым алгоритмом, то есть если существует алгоритм, который печатает (в произвольном порядке и с произвольными промежутками времени) все элементы этого множества и только их.

Такой алгоритм не имеет входа; напечатав несколько чисел, он может надолго задуматься и следующее число напечатать после большого перерыва (а может вообще больше никогда ничего не напечатать — тогда множество будет конечным).

Существует много эквивалентных определений перечислимого множества. Вот некоторые из них:

1) Множество перечислимо, если оно есть область определения вычислимой функции.

2) Множество перечислимо, если оно есть область значений вычислимой функции.

3) Множество X перечислимо, если его (как иногда говорят) «полухарактеристическая» функция, равная 0 на элементах X и не определённая вне X, вычислима.

Чтобы доказать эквивалентность этих определений, воспользуемся возможностью пошагового исполнения алгоритма.

Пусть X перечисляется некоторым алгоритмом A. Покажем, что полухарактеристическая функция множества X вычислима. В самом деле, алгоритм её вычисления таков: получив на вход число n, пошагово выполнять алгоритм A, ожидая, пока он напечатает число n. Как только он это сделает, выдать на выход 0 и закончить работу.

Наоборот, пусть X есть область определения (вычислимой) функции f, вычисляемой некоторым алгоритмом B. Тогда X перечисляется таким алгоритмом A:

> *Параллельно запускать B на входах $0, 1, 2, \ldots$, делая всё больше шагов работы алгоритма B (сначала один шаг работы на входах 0 и 1; потом по два шага работы на входах $0, 1, 2$, потом по три на входах $0, 1, 2, 3$ и так далее). Все аргументы, на которых алгоритм B заканчивает работу, печатать по мере обнаружения.*

Итак, мы установили эквивалентность исходного определения определениям 1 и 3. Если в только что приведённом описании алгоритма A печатать не аргументы, на которых B заканчивает работу, а результаты этой работы, то получается алгоритм, перечисляющий область значений функции f. Осталось ещё убедиться, что всякое перечислимое множество есть область значений вычислимой функции. Это можно сделать, например, так: пусть

X есть область определения вычислимой функции, вычисляемой некоторым алгоритмом A. Тогда X есть область значений функции

$$b(x) = \begin{cases} x, & \text{если } A \text{ заканчивает работу на } x, \\ \text{не определено в противном случае.} \end{cases}$$

Вычисляющий эту функцию алгоритм действует так же, как и A, но только вместо результата работы алгоритма A выдаёт копию входа.

Ещё одно эквивалентное определение перечислимого множества: множество натуральных чисел перечислимо, если оно либо пусто, либо есть множество значений всюду определённой вычислимой функции (другими словами, его элементы можно расположить в вычислимую последовательность).

В самом деле, пусть перечислимое множество X, перечисляемое алгоритмом A, непусто. Возьмём в нём какой-то элемент x_0. Теперь рассмотрим такую всюду определённую функцию a: если на n-м шаге работы алгоритма A появляется число t, то положим $a(n) = t$; если же ничего не появляется, то положим $a(n) = x_0$. (Мы предполагаем, что на данном шаге работы алгоритма может появиться только одно число — в противном случае работу надо разбить на более мелкие шаги.)

Заметим, что это рассуждение неконструктивно — имея алгоритм A, мы можем не знать, пусто ли перечисляемое им множество или нет.

Теорема 1. Пересечение и объединение перечислимых множеств перечислимы.

$\triangleleft$ Если X и Y перечисляются алгоритмами A и B, то их объединение перечисляется алгоритмом, который параллельно выполняет по шагам A и B и печатает всё, что печатают A и B. С пересечением немного сложнее — результаты работы A и B надо накапливать и сверять друг с другом; что появится общего — печатать. $\triangleright$

4. Проведите это рассуждение, используя какое-либо другое эквивалентное определение перечислимости.

Как мы увидим, дополнение перечислимого множества не обязано быть перечислимым.

5. Иногда говорят о так называемых «недетерминированных алгоритмах» (оксюморон, но распространённый) — такой алгоритм включает в себя команды типа

`n :=` произвольное натуральное число

(достаточно, впрочем, команды «`n :=` 0 или 1», так как произвольное число можно формировать по битам). Недетерминированный алгоритм (при одном и том же входе) может действовать по-разному, в зависимости от того, какие «произвольные» числа будут выбраны. Докажите, что перечислимое множество можно эквивалентно определить как множество чисел, которые могут появиться на выходе недетерминированного алгоритма (при фиксированном входе).

6. Докажите, что если множества $A \subset \mathbb{N}$ и $B \subset \mathbb{N}$ перечислимы, то их декартово произведение $A \times B \subset \mathbb{N} \times \mathbb{N}$ также перечислимо.

1.4. Перечислимые и разрешимые множества

Теорема 2. Всякое разрешимое множество натуральных чисел перечислимо. Если множество A и его дополнение (до множества всех натуральных чисел) перечислимы, то A разрешимо.

$\triangleleft$ Если принадлежность числа к множеству A можно проверить некоторым алгоритмом, то A и его дополнение перечислимы: надо по очереди проверять принадлежность чисел $0, 1, 2, \ldots$ и печатать те из них, которые принадлежат A (или те, которые не принадлежат A).

В другую сторону: если у нас есть алгоритм, перечисляющий A, а также другой алгоритм, перечисляющий дополнение к A, то для выяснения принадлежности заданного числа n к A надо запустить оба эти алгоритма и ждать, пока один из них напечатает n (мы знаем, что рано или поздно ровно один из них это сделает). Посмотрев, какой алгоритм это сделал, мы узнаем, лежит ли n в A. $\triangleright$

Этот факт называют *теоремой По́ста*.

Она говорит, что разрешимые множества — это перечислимые множества с перечислимыми дополнениями. Напротив, перечислимые множества можно определить через разрешимые:

Теорема 3. Множество P натуральных чисел перечислимо тогда и только тогда, когда оно является проекцией некоторого разрешимого множества Q пар натуральных чисел. (Проекция получается, если от пар оставить их первые компоненты: $x \in P \Leftrightarrow \exists y(\langle x, y\rangle \in Q)$.)

$\lhd$ Проекция любого перечислимого множества перечислима (перечисляющий алгоритм должен лишь удалять вторые члены пар), так что проекция разрешимого множества тем более перечислима.

Напротив, если P — перечислимое множество, перечисляемое алгоритмом A, то оно есть проекция разрешимого множества Q, состоящего из всех таких пар $\langle x, n\rangle$, что x появляется в течении первых n шагов работы алгоритма A. (Это свойство, очевидно, разрешимо.) $\rhd$

1.5. Перечислимость и вычислимость

Мы видели, что перечислимое множество можно определить в терминах вычислимых функций (например, как область определения вычислимой функции). Можно сделать и наоборот:

Теорема 4. Функция f с натуральными аргументами и значениями вычислима тогда и только тогда, когда её график

$$F = \{\langle x, y\rangle \mid f(x) \text{ определено и равно } y\}$$

является перечислимым множеством пар натуральных чисел.

$\lhd$ Пусть f вычислима. Тогда существует алгоритм, перечисляющий её область определения, то есть печатающий все x, на которых f определена. Если теперь для каждого из таких x вычислять ещё и значение $f(x)$, получим алгоритм, перечисляющий множество F.

Напротив, если имеется алгоритм, перечисляющий F, то функция f вычисляется таким алгоритмом: имея на входе n, ждём появления в F пары, первый член которой равен n; как только такая пара появилась, печатаем её второй член и кончаем работу. $\rhd$

Пусть f — частичная функция с натуральными аргументами и значениями. *Образ* множества A при f определяется как множество всех чисел $f(n)$, для которых $n \in A$ и $f(n)$ определено. *Прообраз* множества A при f определяется как множество всех тех n, при которых $f(n)$ определено и принадлежит A.

Теорема 5. Прообраз и образ перечислимого множества при вычислимой функции перечислимы.

$\triangleleft$ В самом деле, прообраз перечислимого множества A при вычислимой функции f можно получить так: взять график f, пересечь его с перечислимым множеством $\mathbb{N} \times A$ и спроектировать на первую координату. Рассуждение для образов аналогично, только координаты меняются местами. $\triangleright$

7. Пусть F — перечислимое множество пар натуральных чисел. Докажите. что существует вычислимая функция f, определённая на тех и только тех x, для которых найдётся y, при котором $\langle x, y\rangle \in F$, причём значение $f(x)$ является одним из таких y. (Это утверждение называют иногда *теоремой об униформизации*.)

8. Даны два пересекающихся перечислимых множества X и Y. Докажите, что найдутся непересекающиеся перечислимые множества $X' \subset X$ и $Y' \subset Y$, для которых $X' \cup Y' = X \cup Y$.

9. *Диофантовым* называется уравнение, имеющее вид $P(x_1, \ldots, x_n) = 0$, где P — многочлен с целыми коэффициентами. Докажите, что множество диофантовых уравнений, имеющих целые решения, перечислимо. (Оно неразрешимо: в этом состоит известный результат Ю. В. Матиясевича, явившийся решением знаменитой «10-й проблемы Гильберта».)

10. Не ссылаясь на доказательство теоремы Ферма, покажите, что множество всех показателей n, для которых существует решение уравнения $x^n + y^n = z^n$ в целых положительных числах, перечислимо. (Как теперь известно, это множество содержит лишь числа 1 и 2.)

11. Покажите, что всякое бесконечное перечислимое множество можно записать в виде $\{a(0), a(1), a(2), \ldots\}$, где a — вычислимая функция, все значения которой различны. (Указание: в ходе перечисления удаляем повторения.)

12. Покажите, что всякое бесконечное перечислимое множество содержит бесконечное разрешимое подмножество. (Указание: воспользуемся предыдущей задачей и выберем возрастающую подпоследовательность.)

13. Покажите, что для всякой вычислимой функции f существует вычислимая функция, являющаяся «псевдообратной» к f в следующем смысле: область определения g совпадает с областью значений f, и при этом $f(g(f(x))) = f(x)$ для всех x, при которых $f(x)$ определено.

14. Действительное число α называется *вычислимым*, если существует вычислимая функция a, которая по любому рациональному $\varepsilon > 0$ даёт рациональное приближение к α с ошибкой не более ε, т. е. $|\alpha - a(\varepsilon)| \leqslant \varepsilon$ для любого рационального $\varepsilon > 0$. (Рациональное число является конструктивным объектом, так что понятие вычислимости не требует специального уточнения.)

(**а**) Докажите, что число α вычислимо тогда и только тогда, когда множество рациональных чисел, меньших α, разрешимо.

(**б**) Докажите, что число α вычислимо тогда и только тогда, когда последовательность знаков представляющей его десятичной (или двоичной) дроби вычислима.

(**в**) Докажите, что число α вычислимо тогда и только тогда, когда существует вычислимая последовательность рациональных чисел, вычислимо сходящаяся к α (последнее означает, что можно алгоритмически указать N по ε в стандартном ε-N-определении сходимости.)

(**г**) Покажите, что сумма, произведение, разность и частное вычислимых действительных чисел вычислимы. Покажите, что корень многочлена с вычислимыми коэффициентами вычислим.

(**д**) Сформулируйте и докажите утверждение о том, что предел вычислимо сходящейся последовательности вычислимых действительных чисел вычислим.

(**е**) Действительное число α называют *перечислимым снизу*, если множество всех рациональных чисел, меньших α, перечислимо. (Перечислимость сверху определяется аналогично.) Докажите, что число α перечислимо снизу тогда и только тогда, когда оно является пределом некоторой вычислимой возрастающей последовательности рациональных чисел.

(**ж**) Докажите, что действительное число вычислимо тогда и только тогда, когда оно перечислимо снизу и сверху.

Дальнейшие свойства вычислимых действительных чисел см. в задаче 23.

2. Универсальные функции и неразрешимость

2.1. Универсальные функции

Сейчас мы построим пример перечислимого множества, не являющегося разрешимым. При этом будет использоваться так называемая универсальная функция.

Говорят, что функция U двух натуральных аргументов является *универсальной* для класса вычислимых функций одного аргумента, если для каждого n функция

$$U_n\colon x \mapsto U(n,x)$$

(«сечение» функции U при фиксированном n) является вычислимой и если все вычислимые функции (одного аргумента) встречаются среди U_n. (Напомним, что ни функция U, ни вычислимые функции одного аргумента не обязаны быть всюду определёнными.)

Аналогичное определение можно дать и для других классов функций (одного аргумента): например, функция U двух аргументов будет универсальной для класса всех всюду определённых вычислимых функций одного аргумента, если её сечения U_n являются всюду определёнными вычислимыми функциями одного аргумента и исчерпывают все такие функции. Очевидно, универсальные функции существуют для любых счётных классов (и только для них).

Ключевую роль в этом разделе играет такой факт:

Теорема 6. Существует вычислимая функция двух аргументов, являющаяся универсальной функцией для класса вычислимых функций одного аргумента.

$\lhd$ Запишем все программы, вычисляющие функции одного аргумента, в вычислимую последовательность $p_0, p_1, \ldots$ (например, в порядке возрастания их длины). Положим $U(i,x)$ равным результату работы i-ой программы на входе x. Тогда функция U и будет искомой вычислимой универсальной функцией. Сечение U_i будет вычислимой функцией, вычисляемой программой p_i.

Алгоритм, вычисляющий саму функцию U, есть по существу интерпретатор для используемого языка программирования (он применяет первый аргумент ко второму, если отождествить программу и её номер). $\triangleright$

15. Все сечения U_n некоторой функции U двух аргументов вычислимы. Следует ли отсюда, что функция U вычислима?

16. Дайте (естественное) определение понятия вычислимой функции трёх аргументов, универсальной для класса вычислимых функций двух аргументов, и докажите её существование.

Для множеств используется аналогичная терминология: множество $W \subset \mathbb{N} \times \mathbb{N}$ называют *универсальным* для некоторого класса множеств натуральных чисел, если все сечения

$$W_n = \{x \mid \langle n, x \rangle \in W\}$$

множества W принадлежат этому классу и других множеств в классе нет.

Теорема 7. Существует перечислимое множество пар натуральных чисел, универсальное для класса всех перечислимых множеств натуральных чисел.

$\triangleleft$ Рассмотрим область определения универсальной функции U. Она будет универсальным перечислимым множеством, поскольку всякое перечислимое множество является областью определения некоторой вычислимой функции U_n. $\triangleright$

17. Как построить универсальное множество, исходя из того, что всякое перечислимое множество есть множество значений некоторой функции U_n?

18. Существует ли разрешимое множество пар натуральных чисел, универсальное для класса всех разрешимых множеств натуральных чисел?

2.2. Диагональная конструкция

В предыдущем разделе мы построили универсальную функцию для класса всех вычислимых функций одного аргумента. Можно ли сделать то же самое для класса всюду определённых вычислимых функций? Оказывается, что нет.

Теорема 8. Не существует вычислимой всюду определённой функции двух аргументов, универсальной для класса всех вычислимых всюду определённых функций одного аргумента.

$\lhd$ Воспользуемся «диагональной конструкцией» — точно так же доказывается несчётность множества всех бесконечных десятичных дробей. Пусть U — произвольная вычислимая всюду определённая функция двух аргументов. Рассмотрим диагональную функцию $u(n) = U(n, n)$. Очевидно, на аргументе n функция u совпадает с функцией U_n, а функция $d(n) = u(n) + 1$ отличается от U_n. Таким образом, вычислимая всюду определённая функция $d(n)$ отличается от всех сечений U_n, и потому функция U не является универсальной. $\rhd$

Почему это рассуждение не проходит для класса всех вычислимых функций (в том числе частичных)? Дело в том, что значение $d(n) = U(n, n) + 1$ теперь не обязано отличаться от значения $U_n(n) = U(n, n)$, так как оба они могут быть не определены.

Тем не менее, часть рассуждения остаётся в силе.

Теорема 9. Существует вычислимая функция d (с натуральными аргументами и значениями), от которой никакая вычислимая функция f не может всюду отличаться: для любой вычислимой функции f найдётся такое число n, что $f(n) = d(n)$ (последнее равенство понимается в том смысле, что либо оба значения $f(n)$ и $d(n)$ не определены, либо оба определены и равны).

$\lhd$ По существу всё уже сказано: такова диагональная функция $d(n) = U(n, n)$ (здесь U — вычислимая функция двух аргументов, универсальная для класса вычислимых функций одного аргумента). Любая вычислимая функция f есть U_n при некотором n и потому $f(n) = U_n(n) = U(n, n) = d(n)$. $\rhd$

Теорема 10. Существует вычислимая функция, не имеющая всюду определённого вычислимого продолжения.

$\lhd$ Такова, например, функция $d'(n) = d(n)+1$, где d — функция из предыдущей теоремы. В самом деле, любое её всюду определённое продолжение всюду отличается от d (в тех местах, где функция d определена, функция d' на единицу больше d и потому любое продолжение функции d' отличается от d; там, где d не определена, любая всюду определённая функция отличается от d). $\triangleright$

19. Докажите, что и сама функция d из доказательства предыдущей теоремы не имеет вычислимого всюду определённого продолжения.

2.3. Перечислимое неразрешимое множество

Теперь мы можем доказать обещанное утверждение.

Теорема 11. Существует перечислимое неразрешимое множество. (Переформулировка: существует перечислимое множество с неперечислимым дополнением.)

$\triangleleft$ Рассмотрим вычислимую функцию $f(x)$, не имеющую всюду определённого вычислимого продолжения. Её область определения F будет искомым множеством. В самом деле, F перечислимо (по одному из определений перечислимости). Если бы F было разрешимо, то функция

$$g(x) = \begin{cases} f(x), & \text{если } x \in F, \\ 0, & \text{если } x \notin F \end{cases}$$

была бы вычислимым всюду определённым продолжением функции f (при вычислении $g(x)$ мы сначала проверяем, лежит ли x в F, если лежит, то вычисляем $f(x)$). $\triangleright$

Полезно проследить, какое именно множество в итоге оказалось перечислимым и неразрешимым. Легко понять, что это множество тех n, при которых $U(n,n)$ определено. Если вспомнить конструкцию функции U, то это множество тех n, при которых n-я программа останавливается на n. Поэтому иногда говорят, что «проблема самоприменимости» (применимости программы к своему номеру) неразрешима.

Заметим, что отсюда следует, что и область определения всей универсальной функции U является перечислимым неразрешимым множеством пар. (Если бы проблема выяснения применимости программы к произвольному аргументу была бы разрешима, то и её частный случай — применимость программы к себе — был бы разрешим.)

20. Пусть U — перечислимое множество пар натуральных чисел, универсальное для класса всех перечислимых множеств натуральных чисел. Докажите, что его «диагональное сечение» $K = \{x \mid \langle x, x\rangle \in U\}$ является перечислимым неразрешимым множеством.

21. Некоторое множество S натуральных чисел разрешимо. Разложим все числа из S на простые множители и составим множество D всех простых чисел, встречающихся в этих разложениях. Можно ли утверждать, что множество D разрешимо?

22. Множество $U \subset \mathbb{N} \times \mathbb{N}$ разрешимо. Можно ли утверждать, что множество «нижних точек» множества U, то есть множество

$$V = \{\langle x, y\rangle \mid \ (\langle x, y\rangle \in U) \text{ и } (\langle x, z\rangle \notin U \text{ для всех } z < y)\}$$

является разрешимым? Можно ли утверждать, что V перечислимо, если U перечислимо?

23. Покажите, что существуют перечислимые снизу, но не вычислимые числа в смысле определений, данных на с. 16. (Указание. Рассмотрим сумму ряда $\sum 2^{-k}$ по всем k из какого-либо перечислимого множества P. Она всегда перечислима снизу, но будет вычислимой только при разрешимом P.)

Мы вернёмся к вычислимым действительным числам в задачах 31 и 60.

2.4. Перечислимые неотделимые множества

Небольшая модификация рассуждения позволяет доказать усиление доказанной выше теоремы:

Теорема 12. Существует вычислимая функция, принимающая только значения 0 и 1 и не имеющая всюду определённого вычислимого продолжения.

$\triangleleft$ Вместо функции $d'(x) = d(x)+1$ можно рассмотреть функцию

$$d''(x) = \begin{cases} 1, & \text{если } d(x) = 0, \\ 0, & \text{если } d(x) > 0 \end{cases}$$

(имеется в виду, что $d''(x)$ не определено, если $d(x)$ не определено). Тогда любое всюду определённое продолжение функции d'' будет по-прежнему отличаться от d всюду и потому не будет вычислимым. $\triangleright$

Этот результат можно перевести на язык перечислимых множеств. Говорят, что два непересекающихся множества X и Y *отделяются* множеством C, если множество C содержит одно из них и не пересекается с другим.

Теорема 13. Существуют два непересекающихся перечислимых множества X и Y, которые не отделяются никаким разрешимым множеством.

◁ В самом деле, пусть d — вычислимая функция, принимающая только значения 0 и 1 и не имеющая всюду определённого вычислимого продолжения. Пусть $X = \{x \mid d(x) = 1\}$ и $Y = \{x \mid d(x) = 0\}$. Легко видеть, что множества X и Y перечислимы. Пусть они отделяются разрешимым множеством C; будем считать, что C содержит X и не пересекается с Y (если наоборот, перейдём к дополнению). Тогда характеристическая функция множества C (равная 1 внутри C и 0 вне него) продолжает d. ▷

Заметим, что этот результат усиливает утверждение о существовании перечислимого неразрешимого множества (если два множества не отделимы разрешимыми множествами, то ни одно из них не разрешимо).

24. Как описать построенные перечислимые неотделимые множества в терминах универсальной функции $U(n, x)$?

25. Покажите, что существует счётное число непересекающихся перечислимых множеств, никакие два из которых нельзя отделить разрешимым множеством.

2.5. Простые множества: конструкция Поста

Существуют и другие конструкции перечислимых неразрешимых множеств. Вот одна из них (предложенная Э. Постом).

Назовём множество *иммунным*, если оно бесконечно, но не содержит бесконечных перечислимых подмножеств. Перечислимое множество называют *простым*, если его дополнение иммунно. (Очевидно, такое множество не может быть разрешимым.)

Теорема 14. Существует простое множество.

◁ Нам нужно, чтобы перечислимое множество S имело иммунное дополнение. Это означает, что S должно пересекаться с любым бесконечным перечислимым множеством. Чтобы гарантировать это, полезно для каждого перечислимого множества V добавить какой-то его элемент в S (хотя бы для бесконечных V). При этом надо позаботиться о том, чтобы вне S осталось бесконечно много элементов. Это можно гарантировать, если добавлять достаточно больши́е элементы (например, из множества номер i добавлять только один элемент, притом бо́льший $2i$).

Объясним конструкцию подробнее. Пусть W — универсальное перечислимое множество пар, среди сечений W_i которого встречаются все перечислимые множества натуральных чисел. Будем называть W_i «перечислимым множеством номер i» (при этом разным номерам может соответствовать одно и то же множество). Рассмотрим множество пар $T = \{\langle i, x\rangle \mid (x \in W_i)$ и $(x > 2i)\}$. Это множество перечислимо (как пересечение W и разрешимого множества $\{\langle i, x\rangle \mid x > 2i\}$). Перечисляя его, будем отбрасывать пары, у которых первый член уже встречался ранее. Останется некоторое перечислимое подмножество $T' \subset T$. Рассмотрим теперь перечислимое множество S вторых членов пар, входящих в T'.

Это множество пересекается с любым бесконечным перечислимым множеством. В самом деле, если W_i бесконечно, то оно содержит и числа, бо́льшие $2i$, поэтому в T (а, значит, и в T') есть пары с первым членом i. Второй член такой пары из T' будет лежать и в S, и в W_i.

С другой стороны, множество S имеет бесконечное дополнение, поскольку среди чисел от 0 до $2n-1$ максимум n различных чисел могут принадлежать S (это числа, попавшие в S с одной из первых n вертикалей — все остальные будут уже больше $2n$). $\triangleright$

26. Докажите, что бесконечное множество, не содержащее бесконечных разрешимых подмножеств, иммунно.

27. Докажите, что существует перечислимое множество, для которого прямой пересчёт (последовательность элементов в порядке возрастания без повторений) его дополнения не ограничен сверху никакой всюду определённой вычислимой функцией. Докажите, что это множество является простым.

3. Нумерации и операции

3.1. Главные универсальные функции

Очевидно, композиция двух вычислимых функций вычислима. При этом это утверждение кажется «эффективным» в том смысле, что по программам двух функций можно алгоритмически получить программу их композиции. В разумном языке программирования она будет состоять из двух подпрограмм, соответствующих двум вычислимым функциям, и главной программы с единственной строкой «`return (f(g(x)))`».

Однако мы хотим говорить не о программах (чтобы не вдаваться в детали языка программирования), а о номерах функций. Для этого у нас есть средства. Именно, всякая универсальная функция U для класса вычислимых функций одного аргумента задаёт нумерацию этого класса: число n является номером функции $U_n : x \mapsto U(n,x)$.

Вообще *нумерацией* (более точно, *натуральной нумерацией*) произвольного множества $\mathcal{F}$ называют всюду определённое отображение $\nu\colon \mathbb{N} \to \mathcal{F}$, область значений которого есть всё множество $\mathcal{F}$. Если $\nu(n) = f$, то число n называют *номером* объекта f. Таким образом, всякая функция двух аргументов задаёт нумерацию некоторого класса функций одного аргумента (и является универсальной для этого класса).

Наша цель — сформулировать и доказать такое утверждение: (при некоторых условиях на нумерацию вычислимых функций) существует алгоритм, который по любым двум номерам вычислимых функций даёт некоторый номер их композиции.

Прежде всего мы потребуем, чтобы универсальная функция, задающая нумерацию, была вычислимой. (Такие нумерации называют *вычислимыми*.) Однако этого условия мало: нам потребуется, чтобы нумерация была, как говорят, *главной* (*гёделевой*).

Пусть U — двуместная вычислимая универсальная функция для класса одноместных вычислимых функций (термин «k-местная функция» означает «функция k аргументов»). Её называют *главной* универсальной функцией, если для любой двуместной вычислимой функции V существует всюду определённая вычислимая функция $s(m)$, для которой

$$V(m, x) = U(s(m), x)$$

при всех m и x (равенство понимается, как обычно, в том смысле, что либо оба значения не определены, либо определены и равны).

Другими словами, $V_m = U_{s(m)}$, то есть функция s даёт по V-номеру некоторой функции некоторый U-номер той же функции.

Теорема 15. Существует главная универсальная функция.

$\triangleleft$ (Первый способ.) Покажем, что описанное в доказательстве теоремы 6 (с. 17) построение универсальной функции даёт главную универсальную функцию. Напомним, что мы перечисляли все программы $p_0, p_1, p_2, \ldots$ какого-то естественного языка программирования в порядке возрастания их длин и полагали $U(n, x)$ равным результату применения программы p_n к входу x. Пусть теперь есть какая-то другая вычислимая функция V двух аргументов. Нам надо по любому натуральному m получить программу функции V_m, то есть функции, которая получится, если в V зафиксировать первый аргумент равным m. Ясно, что такую программу (в большинстве языков программирования) получить легко — надо только в программе для V заменить первый аргумент на определение константы (или использовать программу для V в качестве подпрограммы, а в основной программе вызывать V с фиксированным первым аргументом).

(Второй способ.) Но можно и не вдаваться в детали построения универсальной функции, а воспользоваться лишь фактом её существования.

Заметим сначала, что существует вычислимая функция трёх аргументов, универсальная для класса вычислимых функций двух аргументов, то есть такая функция T, что при фиксации первого аргумента среди функций $T_n(u,v) = T(n,u,v)$ встречаются все вычислимые функции двух аргументов.

Такую функцию можно построить так. Фиксируем некоторую вычислимую нумерацию пар, то есть вычислимое взаимно однозначное соответствие $\langle u,v\rangle \leftrightarrow [u,v]$ между $\mathbb{N}\times\mathbb{N}$ и $\mathbb{N}$; число $[u,v]$, соответствующее паре $\langle u,v\rangle$, мы будем называть номером этой пары. Если теперь R — двуместная вычислимая универсальная функция для вычислимых одноместных функций, то вычислимая функция T, определённая формулой $T(n,u,v) = R(n,[u,v])$, будет универсальной для вычислимых двуместных функций. В самом деле, пусть F — произвольная вычислимая функция двух аргументов. Рассмотрим вычислимую одноместную функцию f, определённую соотношением $f([u,v]) = F(u,v)$. Поскольку R универсальна, найдётся число n, для которого $R(n,x) = f(x)$ при всех x. Для этого n выполнены равенства $T(n,u,v) = R(n,[u,v]) = f([u,v]) = F(u,v)$, и потому n-ое сечение функции T совпадает с F. Итак, универсальная функция трёх аргументов построена.

Теперь используем её для определения главной универсальной функции U двух аргументов. Неформально говоря, мы встроим внутрь U все другие вычислимые функции двух аргументов, и тем самым U станет главной. Формально говоря, положим $U([n,u],v) = T(n,u,v)$ и проверим, что функция U будет главной. Любая вычислимая функция V двух аргументов встречается среди сечений функции T: можно найти такое n, что $V(u,v) = T(n,u,v)$ для всех u и v. Тогда $V(u,v) = U([n,u],v)$ для всех u и v и потому функция s, определённая формулой $s(u) = [n,u]$, удовлетворяет требованиям из определения главной универсальной функции. $\triangleright$

Нумерации, соответствующие главным универсальным функциям, называют *главными*, или *гёделевыми*.

Теперь уже можно доказать точный вариант утверждения, с которого мы начали.

Теорема 16. Пусть U — двуместная главная универсальная функция для класса вычислимых функций одного аргумента. Тогда существует всюду определённая функция c, которая по номерам p и q двух функций одного аргумента даёт номер $c(p,q)$ их композиции: $U_{c(p,q)}$ есть композиция $U_p \circ U_q$, то есть

$$U(c(p,q),x) = U(p,U(q,x))$$

для всех p, q и x.

$\lhd$ Рассмотрим двуместную вычислимую функцию V, для которой $V([p,q],x) = U(p,U(q,x))$. По определению главной универсальной функции, найдётся такая всюду определённая одноместная вычислимая функция s, что $V(m,x) = U(s(m),x)$ для всех m и x. Тогда $V([p,q],x) = U(s([p,q]),x)$ и потому функция c, определённая соотношением $c(p,q) = s([p,q])$, будет искомой. $\rhd$

Повторим это доказательство неформально. Определение главной универсальной функции требует, чтобы для любого другого языка программирования, для которого имеется вычислимый интерпретатор V, существовал бы вычислимый транслятор s программ этого языка в программы языка U. (Для краткости мы не различаем программы и их номера и рассматриваем число m как U-программу функции U_m.)

Теперь рассмотрим новый способ программирования, при котором пара $\langle p,q\rangle$ объявляется программой композиции функций с U-программами p и q. По условию, такие программы можно алгоритмически транслировать в U-программы, что и требовалось доказать.

Любопытно, что верно и обратное к теореме 16 утверждение:

28. Пусть U — двуместная вычислимая универсальная функция для класса вычислимых функций одного аргумента. Если существует всюду определённая функция, которая по номерам p и q двух функций одного аргумента даёт какой-либо номер их композиции, то функция U является главной. (Указание: покажите, что по k можно алгоритмически получать U-номер функции $x \mapsto [k,x]$.)

Естественный вопрос: существуют ли вычислимые универсальные функции, не являющиеся главными? Мы увидим дальше, что существуют.

29. Изменим определение главной универсальной функции и будем требовать существования «транслятора» s лишь для универсальных вычислимых функций V (а не для любых, как раньше). Покажите, что новое определение эквивалентно старому. (Указание: любую функцию можно искусственно переделать в универсальную, «растворив» в ней любую другую универсальную функцию.)

30. Пусть U — главная универсальная функция. Докажите, что для любой вычислимой функции $V(m,n,x)$ существует такая всюду определённая вычислимая функция $s(m,n)$, что $V(m,n,x) = U(s(m,n),x)$ при всех m, n и x. (Указание: объединить m и n в пару.)

3.2. Вычислимые последовательности вычислимых функций

Пусть дана некоторая последовательность $f_0, f_1, \ldots$ вычислимых функций одного аргумента. Мы хотим придать смысл выражению «последовательность $i \mapsto f_i$ вычислима». Это можно сделать двумя способами:

- можно называть эту последовательность вычислимой, если функция F двух аргументов, заданная формулой $F(i,n) = f_i(n)$, является вычислимой.
- можно называть эту последовательность вычислимой, если существует вычислимая последовательность чисел $c_0, c_1, \ldots$, для которой c_i является одним из номеров функции f_i.

Второе определение (в отличие от первого) зависит от выбора нумерации.

Теорема 17. Если нумерация является вычислимой (то есть соответствующая универсальная функция вычислима), то из второго определения следует первое. Если нумерация является к тому же главной, то из первого определения следует второе.

(Впредь, говоря о вычислимой последовательности вычислимых функций, мы будем всегда предполагать, что нумерация является главной, так что можно пользоваться любым из двух определений.)

$\lhd$ Если U — вычислимая универсальная функция, а последовательность $i \mapsto c_i$ вычислима, то функция $F\colon \langle i, x\rangle \mapsto f_i(x) = U(c_i, x)$ вычислима как результат подстановки одной вычислимой функции в другую.

Напротив, если функция F вычислима, а универсальная функция U является главной, то функция-транслятор, существующая по определению главной универсальной функции, как раз и даёт по i один из номеров функции f_i. $\rhd$

31. Пусть фиксирована главная универсальная функция для класса вычислимых функций одного аргумента. Тогда возникает нумерация вычислимых действительных чисел в соответствии с определением на с. 16: номером числа α является любой номер любой функции, которая по рациональному $\varepsilon > 0$ даёт ε-приближение к α.

(**а**) Покажите, что существует алгоритм, который по любым двум номерам двух вычислимых действительных чисел даёт (некоторый) номер их суммы.

(**б**) Покажите, что не существует алгоритма, который по любому номеру любого вычислимого действительного числа отвечает на вопрос, равно ли это число нулю.

(**в**) Как мы видели в задаче 14, всякое вычислимое действительное число имеет вычислимое десятичное разложение. Покажите, что тем не менее нет алгоритма, который по любому номеру любого вычислимого действительного числа даёт номер вычислимой функции, задающей его десятичное разложение.

3.3. Главные универсальные множества

По аналогии с функциями, перечислимое множество $W \subset \mathbb{N}\times\mathbb{N}$ называется *главным универсальным перечислимым множеством* (для класса всех перечислимых подмножеств $\mathbb{N}$), если для любого другого перечислимого множества $V \subset \mathbb{N}\times\mathbb{N}$ найдётся такая всюду определённая вычислимая функция $s\colon \mathbb{N}\to\mathbb{N}$, что

$$\langle n, x\rangle \in V \Leftrightarrow \langle s(n), x\rangle \in W$$

для всех n и x. (Очевидно, что из этого свойства следует универсальность.)

Как и для функций, можно перейти к нумерациям. Каждое множество $U \subset \mathbb{N}\times\mathbb{N}$ задаёт нумерацию некоторого семейства подмножеств натурального ряда: число n является номером n-го сечения $U_n = \{x \mid \langle n, x\rangle \in U\}$. Перечислимое подмножество множества $\mathbb{N}\times\mathbb{N}$ задаёт нумерацию некоторого семейства перечислимых подмножеств натурального ряда; такие нумерации называют *вычислимыми*. Перечислимое множество $W \subset \mathbb{N}\times\mathbb{N}$ универсально, если и только если всякое перечислимое подмножество натурального ряда имеет W-номер; оно является главным тогда и только тогда, когда любая вычислимая нумерация V (любого семейства перечислимых множеств) вычислимо сводится к W-нумерации в том смысле, что $V_n = W_{s(n)}$ для некоторой вычислимой функции s и для всех n.

Теорема 18. Существует главное универсальное перечислимое множество $W \subset \mathbb{N}\times\mathbb{N}$.

$\lhd$ Эта теорема является очевидным следствием такого утверждения:

Лемма. Область определения главной универсальной функции для класса вычислимых функций одного аргумента является главным универсальным множеством для класса перечислимых подмножеств $\mathbb{N}$.

Доказательство леммы. Пусть U — главная универсальная функция, а W — область её определения. Пусть $V \subset \mathbb{N}\times\mathbb{N}$ — произвольное перечислимое множество. Рассмотрим вычислимую функцию G с областью определения V. Поскольку функция U является главной, найдётся всюду определённая вычислимая функция $s\colon \mathbb{N}\to\mathbb{N}$, для которой $G_n = U_{s(n)}$ при всех n. Тогда равны и области определения функций G_n и $U_{s(n)}$, то есть $V_n = W_{s(n)}$. $\rhd$

32. Постройте главное универсальное множество непосредственно, используя универсальное подмножество $\mathbb{N}^3$ (по аналогии с выше приведённым построением главной универсальной функции).

Как и для функций, мы теперь можем доказывать, что различным операциям над множествами соответствуют вычислимые преобразования номеров. Вот лишь один пример такого рода:

Теорема 19. Пусть $W \subset \mathbb{N} \times \mathbb{N}$ — главное универсальное перечислимое множество. Тогда по W-номерам двух перечислимых множеств можно алгоритмически получить номер их пересечения: существует такая вычислимая всюду определённая функция двух аргументов s, что

$$W_{s(m,n)} = W_m \cap W_n$$

для любых двух m и n.

$\triangleleft$ Рассмотрим множество $V \subset \mathbb{N} \times \mathbb{N}$, определённое так:

$$\langle [m,n], x \rangle \in V \Leftrightarrow x \in (W_m \cap W_n)$$

(здесь квадратные скобки обозначают номер пары) и применим к нему определение главного универсального множества. $\triangleright$

Как и для функций, понятие вычислимости последовательности перечислимых множеств может быть определено двояко: можно считать вычислимой последовательность $V_0, V_1, \ldots$ сечений произвольного перечислимого множества V, а можно требовать, чтобы по i можно было алгоритмически указать один из номеров i-го члена последовательности в главной нумерации. Эти определения равносильны (доказательство полностью аналогично рассуждению для функций).

4. Свойства главных нумераций

4.1. Множества номеров

Начнём с такого примера. Рассмотрим множество номеров нигде не определённой функции для какой-либо главной нумерации. Будет ли оно разрешимо? Другими словами, можно ли по номеру функции в главной нумерации определить, является ли эта функция нигде не определённой?

Прежде чем отвечать на этот вопрос, заметим, что ответ не зависит от того, какая главная нумерация выбрана. В самом деле, если есть две разные главные нумерации, то они, как говорят, «сводятся» друг к другу: по номеру функции в одной нумерации можно алгоритмически получить номер той же функции в другой нумерации. Если бы в одной нумерации можно было бы проверять «нигде-не-определённость» функции, то это можно было бы делать и в другой (применив «функции перехода»).

Следующая теорема показывает, что ответ на исходный вопрос будет отрицательным.

Теорема 20. Пусть U — произвольная главная универсальная функция. Тогда множество тех n, при которых функция U_n является нигде не определённой, неразрешимо.

$\triangleleft$ Используем метод, называемый «сведéнием» — покажем, что если бы это множество было разрешимым, то и вообще любое перечислимое множество было бы разрешимым. (Что, как мы знаем, неверно.)

Пусть K — произвольное перечислимое неразрешимое множество. Рассмотрим такую вычислимую функцию V двух аргументов:

$$V(n,x) = \begin{cases} 0, \text{ если } n \in K, \\ \text{не определено, если } n \notin K. \end{cases}$$

Как видно, второй аргумент этой функции фиктивен, и она по существу совпадает с полухарактеристической функцией множества K от первого аргумента. Очевидно, эта функция имеет сечения двух типов: при $n \in K$ сечение V_n является нулевой функцией, при $n \notin K$ — нигде не определённой функцией.

Так как функция U является главной, существует вычислимая всюду определённая функция s, для которой $V(n,x) = U(s(n),x)$ при всех n и x, т. е. $V_n = U_{s(n)}$. Поэтому при $n \in K$ значение $s(n)$ является U-номером нулевой функции, а при $n \notin K$ значение $s(n)$ является U-номером нигде не определённой функции. Поэтому если бы множество U-номеров нигде не определённой функции разрешалось бы некоторым алгоритмом, то мы бы могли применить этот алгоритм к $s(n)$ и узнать, принадлежит ли число n множеству K или нет. Таким образом, множество K было бы разрешимым в противоречии с нашим предположением. $\triangleright$

В частности, мы можем заключить, что нигде не определённая функция имеет бесконечно много номеров в любой главной нумерации (поскольку любое конечное множество разрешимо).

Кроме того, можно заметить, что множество номеров нигде не определённой функции не только не разрешимо, но и не перечислимо. В самом деле, его дополнение — множество всех номеров всех функций с непустой областью определения — перечислимо. (Это верно для любой вычислимой нумерации, а не только для главной: параллельно вычисляя $U(n,x)$ для всех n и x, мы можем печатать те n, для которых обнаружилось x, при котором $U(n,x)$ определено.) А если дополнение неразрешимого множества перечислимо, то само множество неперечислимо (по теореме Поста, с. 13).

Справедливо и более общее утверждение, называемое иногда теоремой Успенского–Райса. Обозначим класс всех вычислимых функций (одного аргумента) через $\mathcal{F}$.

Теорема 21. Пусть $\mathcal{A} \subset \mathcal{F}$ — произвольное нетривиальное свойство вычислимых функций (нетривиальность означает, что есть как функции, ему удовлетворяющие, так и функции, ему не удовлетворяющие, то есть что множество $\mathcal{A}$ непусто и не совпадает со всем $\mathcal{F}$). Пусть U — главная универсальная функция. Тогда не существует алгоритма, который по U-номеру вычислимой функции проверял бы, обладает ли она свойством $\mathcal{A}$. Другими словами, множество $\{n \mid U_n \in \mathcal{A}\}$ неразрешимо.

$\triangleleft$ Посмотрим, принадлежит ли нигде не определённая функция (обозначим её ζ) классу $\mathcal{A}$, и возьмём произвольную функцию ξ «с другой стороны» (если $\zeta \in \mathcal{A}$, то $\xi \notin \mathcal{A}$ и наоборот).

Далее действуем как раньше, но только вместо нулевой функции возьмём функцию ξ: положим

$$V(n,x) = \begin{cases} \xi(x), \text{ если } n \in K, \\ \text{не определено, если } n \notin K. \end{cases}$$

Как и раньше, функция V будет вычислимой (для данных n и x мы ожидаем появления n в множестве K, после чего вычисляем $\xi(x)$). При $n \in K$ функция V_n совпадает с ξ, при $n \notin K$ — с ζ. Таким образом, проверяя свойство $V_n \in \mathcal{A}$ (если бы это можно было сделать вопреки утверждению теоремы), можно было бы узнать, принадлежит ли число n множеству K или нет. $\triangleright$

Некоторым недостатком этого доказательства является его несимметричность (с одной стороны от $\mathcal{A}$ мы берём нигде не определённую функцию, с другой стороны — любую). Вот более симметричный вариант. Покажем, что если свойство $\mathcal{A}$ можно распознавать по U-номерам, то любые два непересекающихся перечислимых множества P и Q отделимы разрешимым множеством. Выберем какие-нибудь две функции ξ и η, находящиеся «по разные стороны» от $\mathcal{A}$. Рассмотрим функцию

$$V(n,x) = \begin{cases} \xi(x), \text{ если } n \in P, \\ \eta(x), \text{ если } n \in Q, \\ \text{не определено, если } n \notin P \cup Q. \end{cases}$$

Эта функция вычислима: для заданных n и x ожидаем, пока n появится либо в P, либо в Q, после чего запускаем вычисление соответственно $\xi(x)$ или $\eta(x)$.

Если $n \in P$, то V_n совпадает с ξ; если $n \in Q$, то V_n совпадает с η. Поэтому, проверяя, принадлежит ли V_n

классу $\mathcal{A}$, мы могли бы разрешимо отделить P от Q. Получаем противоречие, которое и завершает этот более симметричный вариант доказательства.

Второй вариант доказательства показывает, что верно следующее усиление этой теоремы: для любых различных вычислимых функций φ и ψ и любой главной универсальной функции U множества всех U-номеров функции φ и функции ψ не отделимы разрешимым множеством. (Заметим, что эти множества не перечислимы, как мы впоследствии увидим.)

Теперь легко указать пример вычислимой универсальной функции, не являющейся главной. Достаточно сделать так, чтобы нигде не определённая функция имела единственный номер. Это несложно. Пусть $U(n,x)$ — произвольная вычислимая универсальная функция. Рассмотрим множество D всех U-номеров всех функций с непустой областью определения. Как мы уже говорили, это множество перечислимо. Рассмотрим всюду определённую вычислимую функцию d, его перечисляющую: $D = \{d(0), d(1), \ldots\}$. Теперь рассмотрим функцию $V(i,x)$, для которой $V(0,x)$ не определено ни при каком x, а $V(i+1,x) = U(d(i),x)$. Другими словами, функция V_0 нигде не определена, а функция V_{i+1} совпадает с $U_{d(i)}$. Легко понять, что функция V вычислима; она универсальна по построению, и единственным V-номером нигде не определённой функции является число 0.

На самом деле существуют и более экзотические нумерации: как показал Фридберг, можно построить универсальную вычислимую функцию, для которой каждая вычислимая функция будет иметь ровно один номер. Соответствующие нумерации называют *однозначными*; очевидно, они не могут быть главными. Забавная переформулировка: можно разработать такой язык программирования, в котором каждую программистскую задачу можно решить единственным образом. (Доказательство этой теоремы трудно и не приводится, на русском языке оно есть в книжке А. И. Мальцева «Алгоритмы и рекурсивные функции» [5].) Аналогичное утверждение верно и для нумераций перечислимых множеств.

4.2. Новые номера старых функций

Теорема Успенского – Райса показывает, что в главной нумерации множество номеров любой конкретной функции неразрешимо и потому бесконечно. Сейчас мы докажем более сильный факт: по номеру любой функции в главной нумерации можно алгоритмически получить сколько угодно других номеров той же функции. Формально это можно выразить, например, так:

Теорема 22. Пусть U — главная универсальная функция. Тогда существует всюду определённая функция g двух аргументов с таким свойством: для любого i числа $g(i,0), g(i,1), \ldots$ являются различными U-номерами функции U_i.

$\triangleleft$ План таков: мы построим другую систему программирования, в которой заданная функция U_i имеет бесконечно много программ. Затем, пользуясь тем, что функция U — главная, будем «транслировать» эти программы и получать U-номера интересующей нас функции. При этом мы примем специальные меры, чтобы гарантировать, что получится бесконечно много разных номеров (вообще говоря, разные программы после трансляции могут слиться). Объясним это более подробно. Пусть h — произвольная функция. Покажем, что существует алгоритм, отыскивающий бесконечно много различных U-номеров функции h. (В теореме утверждается, что это можно сделать не только для конкретной функции h, но и для всех функций U_i, как говорят, «равномерно по i», но временно забудем про это.)

Пусть P — перечислимое неразрешимое множество. Рассмотрим вычислимую функцию

$$V(n,x) = \begin{cases} h(x), \text{ если } n \in P, \\ \text{не определено, если } n \notin P. \end{cases}$$

Среди функций V_n встречаются всего две: если $n \in P$, то $V_n = h$; если $n \notin P$, то V_n — нигде не определённая функция, которую мы обозначим через ζ. Будем пока считать, что $h \neq \zeta$ (при $h = \zeta$ эта конструкция не работает и нам потребуется чуть более сложная).

Так как U — главная универсальная функция, то существует транслятор s, переводящий V-номера в U-номера. При этом

- $n \in P \Rightarrow U_{s(n)} = V_n = h$;
- $n \notin P \Rightarrow U_{s(n)} = V_n = \zeta$.

Таким образом, если $p(0), p(1), \ldots$ — вычислимое перечисление множества P, то все числа $s(p(0)), s(p(1)), \ldots$ будут U-номерами функции h. Покажем, что среди этих чисел бесконечно много различных (и потому можно вычислять их одно за другим, пока не обнаружится новый, ещё не использованный, номер функции h).

Пусть это не так, и множество $X = \{s(n) \mid n \in P\}$ конечно. Тогда X разрешимо. Если $n \in P$, то $s(n) \in X$ по построению; если $n \notin P$, то $s(n)$ — номер функции ζ и потому не может принадлежать X (напомним, что $h \neq \neq \zeta$). Следовательно, $n \in P$ равносильно $s(n) \in X$, и потому из разрешимости X следует разрешимость P, что противоречит предположению.

Это рассуждение, однако, не проходит, если функция h нигде не определена. Хотя числа $s(p(0)), s(p(1)), \ldots$ по-прежнему будут номерами h, ничто не гарантирует нам, что среди них бесконечно много различных. В этом случае поступим чуть хитрее и рассмотрим любую вычислимую функцию ξ, отличную от ζ (например, тождественно нулевую). Рассмотрим два перечислимых неотделимых множества P и Q и вычислимую функцию

$$V(n,x) = \begin{cases} h(x), \text{ если } n \in P, \\ \xi(x), \text{ если } n \in Q, \\ \text{не определено, если } n \notin P \cup Q. \end{cases}$$

Пусть s — транслятор, переводящий V-номера в U-номера. Тогда

- $n \in P \Rightarrow U_{s(n)} = h$;
- $n \in Q \Rightarrow U_{s(n)} = \xi$;
- $n \notin P \cup Q \Rightarrow U_{s(n)} = \zeta$.

Как и раньше, числа $s(p(0)), s(p(1)), \ldots$ будут номерами функции h. Покажем, что если $h \neq \xi$, то множество X таких чисел неразрешимо (и потому бесконечно). В самом деле, если X разрешимо, то мы можем отделить P от Q разрешимым множеством. Именно, множество $\{n \mid s(n) \in X\}$ содержит P (по построению) и не пересекается с Q (поскольку при $n \in Q$ число $s(n)$ будет номером функции ξ и потому не может лежать в X).

Таким образом, у нас есть две конструкции, позволяющие получать другие номера заданной функции h. Первая из них заведомо годится (даёт бесконечно много новых номеров), если h хотя бы где-то определена; вторая заведомо действует для нигде не определённой h. Пусть нам теперь дана функция h и заранее не известно, является ли она нигде не определённой или нет. Что тогда? Тогда можно применять параллельно обе конструкции, пока одна из них не даст нам требуемый новый номер — мы знаем, что по крайней мере одна из конструкций обязана сработать, хотя и не знаем, какая.

Это позволяет нам провести рассуждение равномерно по i. Формально говоря, надо действовать так. Рассмотрим две вычислимые функции V_1 и V_2 двух аргументов, определённые соотношениями

$$V_1([i,n],x) = \begin{cases} U(i,x), & \text{если } n \in P, \\ \text{не определено}, & \text{если } n \notin P, \end{cases}$$

$$V_2([i,n],x) = \begin{cases} U(i,x), & \text{если } n \in P, \\ 0, & \text{если } n \in Q, \\ \text{не определено}, & \text{если } n \notin P \cup Q \end{cases}$$

(P и Q — фиксированные перечислимые неотделимые множества, $[u,v]$ — номер пары $\langle u,v \rangle$ в фиксированной вычислимой нумерации пар). Поскольку U — главная универсальная функция, можно найти такие вычислимые всюду определённые функции s_1 и s_2, что $V_1([i,n],x) = U(s_1([i,n]),x)$ и $V_2([i,n],x) = U(s_2([i,n]),x)$. Пусть p — всюду определённая функция одного аргумента, для которой $P = \{p(0), p(1), \ldots\}$. Искомую функцию g можно определить так: $g(i, k)$ есть k-ое (без учёта повторений) число в последовательности

$$s_1([i, p(0)]), s_2([i, p(0)]), s_1([i, p(1)]),\\ s_2([i, p(1)]), s_1([i, p(2)]), s_2([i, p(2)]), \ldots \ \triangleright$$

4.3. Изоморфизм главных нумераций

Доказанное только что утверждение о получении новых номеров используется при доказательства теоремы Роджерса об изоморфизме главных (гёделевых) нумераций. Вот её формулировка:

Теорема 23. Пусть U_1 и U_2 — две главные универсальные функции для класса вычислимых функций одного аргумента. Тогда существуют две всюду определённые взаимно обратные вычислимые функции s_{12} и s_{21}, для которых

$$U_1(n, x) = U_2(s_{12}(n), x) \quad \text{и} \quad U_2(n, x) = U_1(s_{21}(n), x)$$

при любых n и x.

Эта теорема показывает, что трансляторы, сводящие одну главную нумерацию к другой и наоборот, можно выбрать взаимно обратными, так что любые две главные нумерации отличаются лишь вычислимой перестановкой номеров. (Заметим, что утверждение предыдущей теоремы 22 является следствием теоремы об изоморфизме главных нумераций. В самом деле, для некоторых главных нумераций — например, для нумераций, получающихся из языков программирования, — есть способы получать сколько угодно новых номеров той же функции, поэтому в силу изоморфизма такой способ есть и для любой другой главной нумерации.)

$\triangleleft$ Мы будем действовать примерно так же, как и при доказательстве изоморфизма счётных плотных всюду упорядоченных множеств без первого и последнего элементов. Построение искомых биекций происходит по шагам. На k-м шаге строится некоторое взаимно однозначное соответствие

$$a_1 \leftrightarrow b_1, a_2 \leftrightarrow b_2, \ldots, a_k \leftrightarrow b_k$$

между двумя конечными k-элементными подмножествами натурального ряда. При этом для каждого i числа a_i и b_i являются номерами одной и той же функции, только в разных нумерациях (a_i — относительно U_1, а b_i — относительно U_2).

На каждом шаге построения мы добавляем новую пару $a_k \leftrightarrow b_k$ с сохранением указанного свойства. При этом постепенно с обеих сторон мы включим все натуральные числа. Тем самым мы получим искомое взаимно однозначное соответствие, и оно будет вычислимым, раз наша конструкция вычислима.

Итак, как мы добавляем новую пару? На чётных шагах мы берём наименьшее натуральное число u, не входящее в левую сторону соответствия (не встречающееся среди a_i). Оно является U_1-номером некоторой функции. Поскольку нумерация U_2 главная, мы можем получить некоторый U_2-номер той же функции. Обозначим его v. Если v не встречается среди b_i, дело сделано: мы добавляем пару $u \leftrightarrow v$ к нашему соответствию. Если встречается, мы пользуемся теоремой 22 и получаем другие U_2-номера той же функции до тех пор, пока среди них не окажется нового, ещё не встречавшегося среди b_i.

На нечётных шагах мы действуем аналогичным образом, но начинаем с наименьшего числа, не встречающегося среди b_i. $\triangleright$

Замечание для программистов (которое не следует воспринимать серьёзно): поскольку, скажем, паскаль и си можно считать главными нумерациями, то по доказанной теореме существуют не просто трансляторы из паскаля в си и наоборот, а взаимно обратные. (Впрочем, основная часть доказательства при этом не используется, поскольку в паскале и си в программу можно добавлять комментарии, получая сколько угодно эквивалентных программ.)

Аналогичная теорема верна и для главных нумераций перечислимых множеств: любые две такие нумерации изоморфны (отличаются вычислимой перестановкой номеров). Доказательство также проводится аналогично; сначала нужно показать, что в главной нумерации по любому номеру можно алгоритмически получить сколько угодно номеров того же множества. При этом применяются два способа — один годится для всех множеств, кроме пустого, другой для пустого.

33. Проведите это рассуждение подробно.

4.4. Перечислимые свойства функций

Мы уже видели (теорема 21), что никакие (нетривиальные) свойства функций не являются разрешимыми. Но некоторые являются перечислимыми (в том смысле, что множество всех номеров всех функций, обладающих этим свойством, перечислимо). Например, таково свойство функции «быть где-то определённой» (см. выше). Другой пример: свойство «$f(19)$ определено и равно 98».

Имеется довольно простое описание всех перечислимых свойств. Чтобы его сформулировать, нам потребуются некоторые определения.

Будем называть *образцом* функцию с натуральными аргументами и значениями, имеющую конечную область определения. Другими словами, образец есть конечный список пар $\langle \text{аргумент}, \text{значение} \rangle$; образцы можно рассматривать как конструктивные объекты (кодировать двоичными словами, натуральными числами и т. п.). Это позволяет говорить о разрешимом множестве образцов, перечислимом множестве образцов и т. п.

Каждому образцу t соответствует свойство функции «продолжать этот образец», то есть множество $\Gamma(t)$ всех (вычислимых) функций, которые являются продолжениями t. (Заметим в скобках, что множества $\Gamma(t)$ образуют базу топологии на множестве всех вычислимых функций.) Легко проверить, что для любого t и для любой вычислимой нумерации U множество всех номеров всех функций из $\Gamma(t)$ перечислимо. В самом деле, надо параллельно вычислять значения $U_n(x)$ для всех n и x; как только накопленные данные позволяют утверждать, что $U_n \in \Gamma(t)$, надо печатать n. (Если $U_n \in \Gamma(t)$, то это обнаружится на конечном шаге, так как область определения образца t конечна.)

Пусть T — произвольное множество образцов. Через $\Gamma(T)$ обозначим множество всех вычислимых функций, которые продолжают хотя бы один образец из T, то есть объединение множеств $\Gamma(t)$ по всем $t \in T$. Теперь мы можем сформулировать обещанный результат:

Теорема 24. (а) Пусть T — произвольное перечислимое множество образцов, а U — вычислимая универсальная функция для класса вычислимых функций одного аргумента. Тогда множество всех U-номеров всех функций из $\Gamma(T)$ перечислимо. (б) Пусть U — главная универсальная функция (для класса вычислимых функций одного аргумента). Пусть $\mathcal{G}$ — некоторое подмножество этого класса. Если множество $\{n \mid U_n \in \mathcal{G}\}$ всех U-номеров всех функций из класса $\mathcal{G}$ перечислимо, то $\mathcal{G} = \Gamma(T)$ для некоторого перечислимого множества образцов T.

Например, упоминавшийся выше класс где-то определённых функций соответствует множеству всех непустых образцов (а также, скажем, множеству всех образцов с одноэлементной областью определения).

$\triangleleft$ Часть (а) утверждения доказывается легко: надо параллельно вычислять все значения $U(n, x)$ и перечислять все образцы из T; как только обнаруживается, что какая-то из функций U_n является продолжением какого-то образца из T, следует печатать n.

Содержательной является часть (б), где мы используем тот факт, что нумерация является главной. Нам понадобятся две леммы.

Лемма 1. Если вычислимая функция h является продолжением вычислимой функции g, принадлежащей классу $\mathcal{G}$, то и функция h принадлежит классу $\mathcal{G}$.

Лемма 2. Если вычислимая функция g принадлежит классу $\mathcal{G}$, то существует функция h с конечной областью определения (образец), принадлежащая классу $\mathcal{G}$, продолжением которой g является.

(Замечание в скобках. В совокупности эти две леммы утверждают, что любое перечислимое свойство $\mathcal{G}$ является открытым в описанной выше топологии.)

Покажем, как из этих лемм вытекает требуемое утверждение. Заметим, что множество T всех образцов, принадлежащих классу $\mathcal{G}$, перечислимо. В самом деле, по образцу как конструктивному объекту (т. е. по списку пар $\langle$аргумент, значение$\rangle$) можно получить его U-номер, поскольку функция U является главной. (Формально: рассмотрим функцию $\langle t, x\rangle \mapsto$ (значение образца t в точке x) и применим определение главной универсальной функции.) Поэтому множество T является перечислимым как прообраз перечислимого множества всех U-номеров всех функций из класса $\mathcal{G}$.

Леммы 1 и 2 гарантируют, что $\mathcal{G} = \Gamma(T)$. В самом деле, по лемме 1 любая функция из $\Gamma(T)$ принадлежит классу $\mathcal{G}$, так как некоторая её часть принадлежит $\mathcal{G}$. С другой стороны, лемма 2 гарантирует, что всякая функция g из $\mathcal{G}$ имеет конечную часть, принадлежащую $\mathcal{G}$ (тем самым и T) и потому $g \in \Gamma(T)$.

Осталось доказать леммы 1 и 2. Пусть, в противоречии с леммой 1, некоторая функция g принадлежит классу $\mathcal{G}$, а её продолжение h — нет. Возьмём перечислимое неразрешимое множество K и рассмотрим такую функцию двух аргументов:

$$V(n, x) = \begin{cases} h(x), \text{ если } n \in K, \\ g(x), \text{ если } n \notin K. \end{cases}$$

Эта функция вычислима. В самом деле, её график, как легко проверить, перечислим как объединение графика g, умноженного на $\mathbb{N}$, и графика h, умноженного на K. Другими словами, чтобы вычислить $V(n, x)$, мы запускаем процесс перечисления K, а также (параллельно с этим процессом) вычисления $g(x)$ и $h(x)$. Результат выдаётся, если завершилось вычисление $g(x)$ (в этом случае уже неважно, лежит ли n в K, так как h есть продолжение g) или если завершилось вычисление $h(x)$ и к тому же n обнаружилось в K.

Поскольку функция U является главной, существует всюду определённая функция s с таким свойством:

- $n \in K \Rightarrow U_{s(n)} = h \Rightarrow U_{s(n)} \notin \mathcal{G}$;
- $n \notin K \Rightarrow U_{s(n)} = g \Rightarrow U_{s(n)} \in \mathcal{G}$.

Тем самым дополнение к K является прообразом перечислимого множества номеров функций из $\mathcal{G}$ при вычислимом отображении s и потому (теорема 5) перечислимо. Полученное противоречие завершает доказательство леммы 1.

Лемма 2 доказывается аналогичным рассуждением. Пусть (вопреки утверждению леммы) функция g принадлежит классу $\mathcal{G}$, но никакая её конечная часть не принадлежит этому классу. Рассмотрим функцию

$$V(n,x) = \begin{cases} g(x), \text{ если после } x \text{ шагов перечисления } K \\ \qquad\qquad\qquad \text{число } n \text{ ещё не появилось,} \\ \text{не определено, если появилось.} \end{cases}$$

Легко видеть, что при $n \notin K$ функция V_n совпадает с g, и потому принадлежит $\mathcal{G}$, а при $n \in K$ функция V_n является конечной частью функции g и потому не принадлежит $\mathcal{G}$. Далее рассуждаем как в лемме 1. $\triangleright$

5. Теорема о неподвижной точке

5.1. Неподвижная точка и отношения эквивалентности

Теорема 25. Пусть U — главная вычислимая универсальная функция для класса вычислимых функций одного аргумента, а h — произвольная всюду определённая вычислимая функция одного аргумента. Тогда существует такое число n, что $U_n = U_{h(n)}$, то есть n и $h(n)$ — номера одной функции.

Другими словами, нельзя найти алгоритма, преобразующего программы, который бы по каждой программе давал другую (не эквивалентную ей). Эту теорему называют *теоремой Клини о неподвижной точке* или *теоремой о рекурсии*.

$\triangleleft$ Мы будем действовать по аналогии с построением вычислимой функции, не имеющей всюду определённого вычислимого продолжения (глава 2).

Рассмотрим произвольное отношение эквивалентности (которое мы будем обозначать $x \equiv y$) на множестве натуральных чисел. Мы покажем, что следующие два свойства этого отношения не могут выполняться одновременно:

- Для всякой вычислимой функции f существует всюду определённая вычислимая функция g, являющаяся её $\equiv$-продолжением (это означает, что если $f(x)$ определено при некотором x, то $g(x) \equiv f(x)$).

- Существует всюду определённая вычислимая функция h, не имеющая $\equiv$-неподвижной точки (то есть функция, для которой $n \not\equiv h(n)$ для всех n).

Если $x \equiv y$ — отношение равенства ($x = y$), то второе свойство выполнено (положим, например, $h(n) = n + 1$), поэтому не выполнено первое. Теорема о неподвижной точке получится, если $x \equiv y$ понимать как $U_x = U_y$ (x и y — номера одной и той же функции). В этом случае выполнено первое свойство, как мы сейчас убедимся, и потому не выполнено второе.

Почему выполнено первое свойство? Пусть f — произвольная вычислимая функция одного аргумента. Рассмотрим функцию $V(n,x) = U(f(n),x)$. Поскольку U является главной универсальной функцией, найдётся всюду определённая функция s, для которой $V(n,x) = U(s(n),x)$ при всех n и x. Эта функция и будет искомым $\equiv$-продолжением. В самом деле, если $f(n)$ определено, то $s(n)$ будет другим номером той же функции, что и $f(n)$. (Отметим, что если $f(n)$ не определено, то $s(n)$ будет одним из номеров нигде не определённой функции.)

Для завершения доказательства теоремы о неподвижной точке осталось проверить, что указанные два свойства отношения эквивалентности несовместны. Это делается так же, как в теореме 9 (раздел 2.2). Возьмём вычислимую функцию f, от которой никакая вычислимая функция не может отличаться всюду (например, диагональную функцию $x \mapsto U(x,x)$ для некоторой вычислимой универсальной функции U). По предположению существует всюду определённое вычислимое $\equiv$-продолжение g функции f. Рассмотрим функцию $t(x) = h(g(x))$, где h — вычислимая всюду определённая функция, не имеющая $\equiv$-неподвижной точки. Тогда t будет всюду отличаться от f. В самом деле, если $f(x)$ определено, то $f(x) \equiv g(x) \not\equiv h(g(x)) = t(x)$, и потому $f(x) \neq t(x)$. Если же $f(x)$ не определено, то этот факт сам по себе уже отличает $f(x)$ и $t(x)$. $\triangleright$

Теорему о неподвижной точке можно переформулировать и так:

Теорема 26. Пусть $U(n,x)$ — главная вычислимая универсальная функция для класса вычислимых функций одного аргумента. Пусть $V(n,x)$ — произвольная вычислимая функция. Тогда функции U и V совпадают на некотором сечении: найдётся такое p, что $U_p = V_p$, то есть $U(p,n) = V(p,n)$ для любого n.

$\triangleleft$ Так как функция U является главной, найдём такую всюду определённую вычислимую функцию h, что $V(n,x) = U(h(n),x)$ при всех n и x. Осталось взять в качестве p неподвижную точку функции h. $\triangleright$

(Пример следствия из этой теоремы: как бы ни старались разработчики, для любых двух версий компилятора существует программа, которая одинаково работает в обеих версиях — например, зацикливается и там, и там. Впрочем, это всё же не наверняка, а только если компилятор задаёт главную универсальную функцию — но надо очень постараться, чтобы это было не так!)

Поучительно развернуть цепочку приведённых рассуждений и проследить, как строится неподвижная точка. Для наглядности вместо $U(n,x)$ мы будем писать $[n](x)$ и читать это «результат применения программы n к входу x»;

Рассуждение начинается с рассмотрения «диагональной» функции $U(x,x)$, которую теперь можно записать как $[x](x)$ (результат применения программы x к себе). Далее мы строим её всюду определённое $\equiv$-продолжение. Это делается так. Выражение $[[x](x)](y)$ вычислимо зависит от двух аргументов. Мы вспоминаем, что U есть главная универсальная функция, и находим такую программу g, что $[[g](x)](y) = [[x](x)](y)$ при всех x и y. При этом $[g](x)$ определено для всех x. Пусть мы хотим найти неподвижную точку программы h. Мы рассматриваем композицию $[h]([g](x))$. Это выражение вычислимо зависит от x, и потому существует программа t, для которой $[t](x) = [h]([g](x))$ при всех x. Эта программа применима ко всем x, поскольку таковы h и g. Теперь неподвижной точкой будет $[g](t)$. Чтобы убедиться в этом, мы должны проверить, что $[[g](t)](x) = [[h]([g](t))](x)$ для всех x. В самом деле, по свойству g имеем $[[g](t)](x) = [[t](t)](x)$. Вспоминая определение t, это выражение можно переписать как $[[h]([g](t))](x)$ — что как раз и требовалось.

5.2. Программа, печатающая свой текст

Классическим примером применения теоремы о неподвижной точке является такое её следствие: существует программа, печатающая (на любом входе) свой собственный текст. В самом деле, если бы такой программы не было, то преобразование

$$p \mapsto \text{(программа, которая на любом входе печатает } p\text{)}$$

не имело бы неподвижной точки.

Формально говоря, это следствие можно выразить так:

Теорема 27. Пусть $U(n,x)$ — главная вычислимая универсальная функция для класса всех вычислимых функций одного аргумента. Тогда существует такое число p, что $U(p,x) = p$ для любого x.

В программистских терминах: пусть $U(p,x)$ — результат применения паскаль-программы p к стандартному входу x. (Уточнения: (1) мы отождествляем числа и последовательности байтов; (2) если программа не завершает работы, мы считаем, что результат не определён, даже если на стандартный выход что-то послано.) Ясно, что функция U будет главной универсальной функцией. Поэтому к ней можно применить сформулированное только что утверждение; получим программу p, которая при любом входе на выходе даёт p.

Ясно, что это рассуждение применимо для любого языка программирования; то, что мы упомянули язык паскаль, роли не играет.

34. Докажите, что существует паскаль-программа, которая печатает свой текст задом наперёд.

35. Покажите, что есть две различные паскаль-программы P и Q с такими свойствами: программа P печатает текст программы Q, а программа Q печатает текст программы P. (Если не требовать различия между P и Q, то в качестве P и Q можно взять одну и ту же программу, печатающую свой текст.)

Выпишем явно программу на паскале, печатающую свой текст. (Это — хорошая задача для любителей программирования.) Для начала напишем неформальную инструкцию на русском языке:

> *напечатать два раза, второй раз в кавычках, такой текст: «напечатать два раза, второй раз в кавычках, такой текст:»*

Чтобы написать что-то похожее на паскале, понадобятся некоторые дополнительные хитрости, но идея ясна: строковая константа используется два раза. Вот один из возможных вариантов:

```
program selfprint;
 var a:array[1..100]of string;i:integer;
begin
a[1]:='program selfprint;';
a[2]:=' var a:array[1..100]of string;i:integer;';
a[3]:='begin';
a[4]:='for i:=1 to 3 do writeln(a[i]);';
a[5]:='for i:=1 to 11 do begin';
a[6]:='  write(chr(97),chr(91),i);';
a[7]:='  write(chr(93),chr(58),chr(61));';
a[8]:='  writeln(chr(39),a[i],chr(39),chr(59));';
a[9]:='end;';
a[10]:='for i:=4 to 11 do writeln(a[i]);';
a[11]:='end.';
for i:=1 to 3 do writeln(a[i]);
for i:=1 to 11 do begin
  write(chr(97),chr(91),i);
  write(chr(93),chr(58),chr(61));
  writeln(chr(39),a[i],chr(39),chr(59));
end;
for i:=4 to 11 do writeln(a[i]);
end.
```

Читая эту программу, полезно иметь в виду соответствие между символами и их кодами:

`a`	`[`	`]`	`:`	`=`	`'`	`;`
97	91	93	58	61	39	59

Видно, что эту программу легко модифицировать, чтобы она, скажем, печатала свой текст задом наперёд — вместо команд `write` и `writeln`, печатающих текст, надо написать команды, записывающие его в файл (или в массив байтов), а потом команды, печатающие этот файл или массив в обратном порядке.

Сделав ещё один шаг, можно получить и доказательство теоремы о неподвижной точке. Пусть h — некоторое преобразование паскаль-программ, у которого мы хотим найти неподвижную точку. Тогда напишем программу наподобие только что приведённой, которая будет записывать свой текст в строку `p`, затем применять h к `p`, получая некоторую другую строку `q`, а затем запускать интерпретатор Паскаля на строке `q` (используя в качестве входа программы `q` вход исходной программы). Конечно, эта программа уже не будет такой короткой, так как будет включать в себя (и даже два раза — первый раз просто так, а второй раз в кавычках) интерпретатор паскаля, написанный на паскале.

Ясно, что такая программа будет неподвижной точкой преобразования h, так как её выполнение начинается ровно с того, что вычисляется значение функции h на её тексте, после чего это значение воспринимается как программа и применяется к входу.

На самом деле это доказательство в сущности повторяет предыдущее, только в более программистских терминах.

5.3. Системный трюк: ещё одно доказательство

Если попросить любителей разных языков программирования написать на своём любимом языке по возможности короткую программу, которая бы печатала свой исходный текст, то чемпионом, скорее всего, окажется короткая программа на бейсике:

```
10 LIST
```

Дело в том, что в бейсике есть команда `LIST`, которая печатает текст программы и может быть запущена изнутри программы.

Прежде всего, это хорошая шутка. Но можно отнестись к ней неожиданно серьёзно и использовать эту идею в ещё одном доказательстве теоремы о неподвижной точке (точнее, в ещё одном варианте того же доказательства).

Прежде всего заметим, что теорему достаточно доказать для какой-то одной главной нумерации. В самом деле, пусть для какой-то другой главной нумерации существует функция без неподвижной точки, то есть имеется способ преобразовывать программы в заведомо неэквивалентные. Тогда с помощью трансляции туда и обратно такой способ можно найти и для первой нумерации (для которой мы теорему считаем доказанной).

Теперь рассмотрим язык программирования, в котором помимо обычных возможностей есть встроенная процедура

```
GetProgramText (var s: string)
```

Эта процедура помещает текст исходной программы в строку `s`. Несмотря на некоторую необычность этой идеи, вполне можно представить себе интерпретатор этого языка — и интерпретация этой процедуры не представляет труда, так как интерпретатору, разумеется, доступен текст программы. Сделаем ещё один шаг и представим себе, что в этом языке есть также процедура

```
ExecuteProgram(s: string)
```

Эта процедура передаёт управление программе, текст которой находится в строке `s`, считая входом этой программы вход исходной программы (как сказал бы настоящий программист, «передавая программе `s` дескриптор входного потока»). И в этом случае понятно, как должен действовать интерпретатор языка: он должен рекурсивно вызвать себя на содержимом строки `s` и входных данных.

Наш обогащённый язык программирования, разумеется, допускает трансляцию с него в обычные языки (поскольку имеет интерпретатор) и наоборот (так как можно не пользоваться новыми конструкциями). Поэтому задаваемая им нумерация вычислимых функций является главной. Пусть h — всюду определённая вычислимая функция, у которой мы хотим найти неподвижную точку. Запишем вычисляющий её алгоритм в виде процедуры нашего языка:

```
function Compute_h (x: string) : string;
begin
  ...
end;
```

(При этом нам даже не нужны новые возможности.) Теперь напишем программу, являющуюся неподвижной точкой функции h:

```
program fixed_point;
   var s: string;
   function Compute_h (x:string) : string;
   begin
     ...
   end;
begin
   GetProgramText (s);
   s := Compute_h (s);
   ExecuteProgram (s);
end.
```

Выполнение этой программы сразу же сводится к выполнению программы, получающейся применением к ней функции h, так что она будет неподвижной точкой по построению.

36. Пусть h — тождественная функция, то есть $h(x) = x$. (Тогда, разумеется, любая программа будет её неподвижной точкой.) Какую именно программу даст нам только что описанная конструкция? (Ответ: программу, которая зацикливается на любом входе.)

Мы только что объяснили, как с помощью языка с дополнительной процедурой «получить текст программы» можно доказать теорему о неподвижной точке. Но можно рассуждать и в обратном направлении и объяснить, почему применение теоремы о неподвижной точке заменяет такую дополнительную процедуру.

В самом деле, пусть мы имеем программу p, в которой есть строка `GetProgramText(s)`. Заменим эту строку на оператор присваивания `s := ` t, где t — некоторая строковая константа. Получится новая программа, зависящая от t. Назовём её $p(t)$. Согласно теореме о неподвижной точке, существует такое значение t, при котором программы t и $p(t)$ эквивалентны. При этом t выполнение программы t эквивалентно выполнению её текста, в котором в момент вызова процедуры `GetProgramText(s)` в строку `s` помещается текст программы t — чего мы и хотели.

Теперь становится понятнее, почему теорема о неподвижной точке называется ещё теоремой о рекурсии. В самом деле, рекурсия состоит в том, что мы вызываем программу изнутри её самой. Здесь происходит даже больше: мы не только имеем право вызвать программу, но и можем получить доступ к её тексту! Обычный вызов действительно является частным случаем доступа к тексту, так как мы можем вызвать процедуру интерпретации на этом тексте. (Конечно, при этом нам понадобится включить в состав программы текст интерпретатора нашего языка программирования, записанный на этом языке.)

5.4. Несколько замечаний

Бесконечное множество неподвижных точек

Теорема 25 (о неподвижной точке) утверждает существование хотя бы одной неподвижной точки. Легко понять, что на самом деле их бесконечно много: в обозначениях этой теоремы существует бесконечно много чисел n, при которых $U_n = U_{h(n)}$

Это можно объяснить, например, так: если бы неподвижных точек было бы конечное число, то можно было бы изменить функцию h в этих точках так, чтобы неподвижных точек не осталось. Недостаток этого рассуждения в том, что оно не позволяет эффективно перечислять неподвижные точки (указать для данной функции h бесконечное перечислимое множество, состоящее из её неподвижных точек). Можно сделать и это, если вспомнить доказательство теоремы 25. В нём неподвижными точками оказывались числа $[g](t)$, а функцию g можно выбрать так, чтобы все её значения были больше любого наперёд заданного числа (теорема 22, с. 36).

37. Проведите это рассуждение подробно.

Неподвижная точка с параметром

Если преобразователь программ вычислимо зависит от некоторого параметра, то и неподвижную точку можно выбрать вычислимо зависящей от этого параметра. Точный смысл этого утверждения таков:

Теорема 28. Пусть U — главная универсальная функция для класса вычислимых функций одного аргумента, а h — всюду определённая вычислимая функция двух аргументов. Тогда существует всюду определённая вычислимая функция n одного аргумента, которая по любому p указывает неподвижную точку для функции h_p, так что $U_{h(p,n(p))} = U_{n(p)}$, или, другими словами,

$$U(h(p, n(p)), x) = U(n(p), x)$$

при всех p и x (как обычно, обе части могут быть одновременно не определены).

$\triangleleft$ Мы видели, что неподвижная точка строится конструктивно. Поэтому если мы ищем неподвижную точку для функции h_p, вычислимо зависящей от параметра p, то и результат нашего построения будет вычислимо зависеть от параметра p.

Конечно, можно было бы формально записать рассуждение, реализующее этот план, но оно довольно громоздко (и вряд ли от этого доказательство станет более понятным). $\triangleright$

В этой теореме мы предполагали, что семейство функций h_p состоит из всюду определённых функций. На самом деле это не обязательно: для произвольного вычислимого семейства вычислимых функций h_p (другими словами, для произвольной вычислимой функции h двух аргументов) существует всюду определённая вычислимая функция n одного аргумента с таким свойством: при каждом p либо функция h_p не определена в точке $n(p)$, либо $n(p)$ является неподвижной точкой функции h_p.

38. Убедитесь, что приведённая в доказательстве теоремы 28 конструкция как раз и даёт функцию $n(p)$ с таким свойством.

39. Объединяя сделанные выше замечания, покажите, что по любой вычислимой функции h (заданной своим номером относительно фиксированной главной универсальной функции) можно эффективно указать бесконечно много чисел, каждое из которых либо будет неподвижной точкой для функции h, либо точкой, где эта функция не определена.

Неподвижная точка для перечислимых множеств

Всё сказанное почти без изменений переносится на главные нумерации перечислимых множеств (если W — главное универсальное перечислимое множество, то всякая вычислимая всюду определённая функция h имеет неподвижную точку n, для которой $W_n = W_{h(n)}$).

В самом деле, если W — главное универсальное перечислимое множество, то к отношению эквивалентности

$$a \equiv b \Leftrightarrow W_a = W_b$$

применимо рассуждение из доказательства теоремы 25, поскольку любая вычислимая функция f имеет вычислимое всюду определённое $\equiv$-продолжение.

Проверим это. Для этого рассмотрим множество

$$V = \{\langle p, x\rangle \mid f(p) \text{ определено и } \langle f(p), x\rangle \in W\}.$$

Легко понять, что это множество перечислимо (например, оно есть область определения вычислимой функции $\langle p, x\rangle \mapsto w(f(p), x)$, где w — вычислимая функция с областью определения W). При этом $V_p = W_{f(p)}$, если $f(p)$ определено, и $V_p = \varnothing$, если $f(p)$ не определено. Вспоминая, что W является главным универсальным множеством, мы находим всюду определённую функцию s, для которой $V_p = W_{s(p)}$. Таким образом, $W_{s(p)} = W_{f(p)}$ для тех p, для которых $f(p)$ определено, что и требовалось.

40. Пусть W — главное универсальное множество (для класса всех перечислимых подмножеств натурального ряда). **(а)** Покажите, что найдётся число x, для которого $W_x = \{x\}$. **(б)** Покажите, что найдутся различные числа x и y, для которых $W_x = \{y\}$ и $W_y = \{x\}$.

Пример использования

Простейшее (хотя не очень типичное) применение теоремы о неподвижной точке — ещё одно доказательство теоремы 21 о неразрешимости свойств вычислимых функций. В самом деле, пусть есть нетривиальное свойство $\mathcal{A}$ вычислимых функций, которое можно распознавать по номерам функций в главной нумерации U. Пусть p — какой-то номер какой-то функции U_p, обладающей этим свойством, а q — какой-то номер какой-то функции U_q, им не обладающим. Тогда функция

$$h(x) = \begin{cases} q, & \text{если функция } U_x \text{ обладает свойством } \mathcal{A}, \\ p, & \text{если функция } U_x \text{ не обладает свойством } \mathcal{A} \end{cases}$$

будет вычислимой и не будет иметь неподвижной точки.

Изоморфизм универсальных множеств

Пусть U_1 и U_2 — два множества пар натуральных чисел. Мы будем называть их *вычислимо изоморфными*, если можно найти вычислимую перестановку (биекцию) $i\colon \mathbb{N} \to \mathbb{N}$ с таким свойством:

$$\langle x, y\rangle \in U_1 \Leftrightarrow \langle i(x), i(y)\rangle \in U_2.$$

Теорема 29. Любые два главных универсальных множества для класса перечислимых подмножеств натурального ряда вычислимо изоморфны.

◁ Для начала объясним разницу между этой теоремой и теоремой об изоморфизме главных нумераций перечислимых множеств (см. замечание в конце раздела 4.3). Там мы применяли вычислимую перестановку только к номерам, но не к элементам множеств. В теперешних обозначениях теорему об изоморфизме главных нумераций можно записать так:

$$\langle x, y\rangle \in U_1 \Leftrightarrow \langle i(x), y\rangle \in U_2.$$

Заметим, что вычислимая перестановка по второму аргументу сохраняет универсальность: если $U \subset \mathbb{N}^2$ —

главное универсальное множество, а $i\colon \mathbb{N} \to \mathbb{N}$ — вычислимая перестановка, то множество пар $\langle x, y\rangle$, для которых $\langle x, i(y)\rangle \in U$, также является главным универсальным множеством. Поэтому из теоремы об изоморфизме главных нумераций перечислимых множеств можно вывести такое следствие: для всякой вычислимой перестановки i найдётся такая вычислимая перестановка i', что

$$\langle x, y\rangle \in U_1 \Leftrightarrow \langle i'(x), i(y)\rangle \in U_2.$$

Если нам повезло и i' совпало с i, то перестановка i будет требуемой. Мы хотим заменить везение ссылкой на теорему о неподвижной точке. На этом пути есть несколько препятствий, но все они преодолимы — и мы сейчас коротко опишем, как.

Прежде всего надо вспомнить доказательство теоремы об изоморфизме главных нумераций и понять, что функция i' (её номер) алгоритмически строится по функции i. Далее мы хотим сослаться на теорему о неподвижной точке, но проблема в том, что это построение работает только для случая, когда функция i является биекцией. Поэтому мы должны модифицировать конструкцию из теоремы об изоморфизме так, чтобы она была применима ко всякой вычислимой функции i и всегда давала бы какую-то биекцию i'; тогда неподвижная точка автоматически будет биекцией.

Сформулируем соответствующее обобщение теоремы об изоморфизме главных нумераций. Дадим предварительно одно вспомогательное определение. Пусть $I\colon \mathbb{N} \to \mathbb{N}$ — произвольная функция. Будем говорить, что множество A *I-соответствует* множеству B, если либо B есть образ A при отображении I, либо A есть прообраз B при этом отображении. (Если I — биекция, эти свойства равносильны.)

Пусть U_1 и U_2 — произвольные главные нумерации перечислимых множеств, а $I\colon \mathbb{N} \to \mathbb{N}$ — вычислимая функция. Тогда существует такая вычислимая биекция $i'\colon \mathbb{N} \to \mathbb{N}$, что при любом k множество с номером k в нумерации U_1 I-соответствует множеству с номером $i'(k)$ в нумерации U_2.

Это обобщение теоремы об изоморфизме главных нумераций доказывается так же, как и сама теорема, и номер функции i' может быть эффективно получен по номеру функции I. Это позволяет применить описанный план и найти биекцию I, при которой $i' = I$, что и требовалось. $\triangleright$

41. Проведите это рассуждение подробно.

Аналогичная теорема верна и для главных универсальных функций.

Теорема 30. Пусть $F_1, F_2 \colon \mathbb{N} \to \mathbb{N}$ — две главные универсальные функции для вычислимых функций одного аргумента. Тогда найдётся такая вычислимая перестановка i, что

$$F_1(x, y) = z \Leftrightarrow F_2(i(x), i(y)) = i(z)$$

для любых натуральных x, y и z.

42. Проведите доказательство этой теоремы по аналогии с предыдущей, используя теорему Роджерса об изоморфизме главных нумераций (с. 39, теорема 23).

6. m-сводимость и свойства перечислимых множеств

6.1. m-сводимость

Мы уже встречались с таким приёмом: чтобы доказать неразрешимость некоторого множества X (например, множества всех номеров всех где-то определённых функции), мы показывали, что если бы оно было разрешимо, то и любое перечислимое множество K было разрешимо. Для этого мы строили вычислимую функцию f так, чтобы принадлежность любого числа n множеству K определялась принадлежностью числа $f(n)$ множеству X.

Сейчас мы изучим такие ситуации более подробно.

Говорят, что множество A натуральных чисел *m-сводится* к другому множеству B натуральных чисел, если существует всюду определённая вычислимая функция $f\colon \mathbb{N} \to \mathbb{N}$ с таким свойством:

$$x \in A \Leftrightarrow f(x) \in B$$

для всех $x \in \mathbb{N}$. Такая функция называется *m-сводящей A к B*. Обозначение: $A \leqslant_m B$.

Теорема 31. (а) Если $A \leqslant_m B$ и B разрешимо, то A разрешимо. (б) Если $A \leqslant_m B$ и B перечислимо, то A перечислимо. (в) $A \leqslant_m A$; если $A \leqslant_m B$ и $B \leqslant_m C$, то $A \leqslant_m C$. (г) Если $A \leqslant_m B$, то $\mathbb{N} \setminus A \leqslant_m \mathbb{N} \setminus B$.

$\lhd$ Все эти свойства почти очевидны. Пусть $A \leqslant_m B$ и мы имеем разрешающий алгоритм для B. Чтобы узнать, принадлежит ли данное x множеству A, мы вычисляем $f(x)$ и узнаём, принадлежит ли $f(x)$ множеству B. Другими словами, $a(x) = b(f(x))$, если a — характеристическая функция множества A, а b — характеристическая функция множества B; поэтому если b вычислима, то и a вычислима как композиция вычислимых функций.

Такое же равенство можно записать для «полухарактеристических» функций, поэтому из перечислимости B

следует перечислимость A. Можно сказать и иначе: множество A является прообразом перечислимого множества B при вычислимом отображении f, и потому перечислимо.

Тождественная функция, очевидно, m-сводит A к A. Если функция f сводит A к B, а функция g сводит B к C, то

$$x \in A \Leftrightarrow f(x) \in B \Leftrightarrow g(f(x)) \in C,$$

так что композиция функций g и f сводит A к C.

Наконец, функция, сводящая A к B, будет сводить и $\mathbb{N} \setminus A$ к $\mathbb{N} \setminus B$. $\triangleright$

Буква «m» в названии исторически происходит из термина «many-one-reducibility»; впрочем, как отмечает М. Сипсер в своём учебнике по теории сложности вычислений, вместо этого лучше говорить «mapping reducibility» (сводимость с помощью отображений), сохраняя букву m в обозначении.

Отметим, что это определение не симметрично относительно перехода к дополнению, если это делается только в одном из множеств: вовсе не обязательно $A \leqslant_m \mathbb{N} \setminus A$, хотя всегда $A \leqslant_m A$.

43. Покажите, что $A \not\leqslant_m \mathbb{N} \setminus A$ для перечислимого неразрешимого множества A.

Отметим, что множества $\varnothing$ и $\mathbb{N}$ являются особыми случаями для m-сводимости. Например, любое разрешимое множество A сводится к любому множеству B, если только B не является пустым и не совпадает с $\mathbb{N}$. В самом деле, если $p \in B$, $q \notin B$ и A разрешимо, то сводящую функцию можно построить так:

$$f(x) = \textbf{if } x \in A \textbf{ then } p \textbf{ else } q \textbf{ fi}.$$

Если же B пусто или совпадает с $\mathbb{N}$, то только пустое множество (соответственно $\mathbb{N}$) сводится к B.

44. Существует ли множество натуральных чисел, к которому m-сводится любое множество натуральных чисел?

6.2. m-полные множества

Теорема 32. Среди перечислимых множеств существуют наибольшие с точки зрения m-сводимости, то есть множества, к которым m-сводится любое перечислимое множество.

$\lhd$ Таковым является универсальное множество (формально надо перейти от пар к их номерам). Пусть $U \subset \subset \mathbb{N} \times \mathbb{N}$ — перечислимое множество пар натуральных чисел, универсальное для класса перечислимых множеств натуральных чисел. Рассмотрим множество V номеров всех пар, входящих в U (для какой-то вычислимой нумерации пар $\langle x, y\rangle \leftrightarrow [x, y] \in \mathbb{N}$). Другими словами,

$$V = \{[x, y] \mid \langle x, y\rangle \in U\}.$$

Пусть T — произвольное перечислимое множество. Тогда $T = U_n$ при некотором n и потому

$$x \in T \Leftrightarrow x \in U_n \Leftrightarrow \langle n, x\rangle \in U \Leftrightarrow [n, x] \in V.$$

Таким образом, функция $x \mapsto [n, x]$ сводит T к V. $\rhd$

Наибольшие относительно m-сводимости перечислимые множества называют *m-полными* (точнее, *m-полными в классе перечислимых множеств*).

Заметим, что если $K \leqslant_m A$ для перечислимых множеств K и A и при этом K является m-полным, то и A является m-полным (в силу транзитивности).

Если универсальное множество является главным, то его диагональ также m-полна:

Теорема 33. Пусть $U \subset \mathbb{N} \times \mathbb{N}$ — главное универсальное множество для класса перечислимых множеств. Тогда его «диагональное сечение» $D = \{x \mid \langle x, x\rangle \in U\}$ является m-полным.

(В частности, множество всех самоприменимых программ является m-полным.)

$\lhd$ Очевидно, D перечислимо. Пусть K — произвольное перечислимое множество. Рассмотрим перечислимое множество пар $V = K \times \mathbb{N}$. Его сечения V_n будут либо пусты (при $n \notin K$), либо совпадать со всем $\mathbb{N}$ (при $n \in \in K$).

Поскольку множество U является главным, существует всюду определённая функция s, для которой $V_n = = U_{s(n)}$. Другими словами, $U_{s(n)}$ совпадает с $\mathbb{N}$ при $n \in K$

и пусто при $n \notin K$. Следовательно, $s(n) \in U_{s(n)}$ (и потому $s(n) \in D$) при $n \in K$ и $s(n) \notin U_{s(n)}$ (и потому $s(n) \notin D$) при $n \notin K$. Таким образом, s сводит K к D. $\triangleright$

45. Докажите, что множество всех программ, останавливающихся на входе 0, является m-полным. Докажите, что множество всех программ, останавливающихся хотя бы на одном входе, является m-полным.

46. Пусть M — m-полное перечислимое множество. Покажите, что существует алгоритм, которой по номеру любой всюду определённой функции h указывает такое число n, что $(n \in M) \Leftrightarrow (h(n) \in M)$. (Указание: это утверждение составляет содержание теоремы о неподвижной точке для некоторого отношения эквивалентности.)

6.3. m-полнота и эффективная неперечислимость

Теория алгоритмов позволяет, как говорят, «конструктивизировать» различные определения. В качестве примера возьмём определение бесконечного множества. Что такое бесконечное множество? Это множество, которое содержит не менее n элементов для любого натурального n. Теперь можно сказать так: множество называется «эффективно бесконечным», если существует алгоритм, который по любому n указывает n различных элементов этого множества.

47. Покажите, что произвольное множество A является эффективно бесконечным тогда и только тогда, когда оно содержит бесконечное перечислимое множество (т. е. не является иммунным, см. с. 22).

Сейчас нас будет интересовать эффективный вариант понятия неперечислимости. Что значит, что множество A неперечислимо? Это означает (трюизм), что A отличается от любого перечислимого множества. Естественно называть множество A эффективно неперечислимым, если по любому перечислимому множеству можно указать место, где оно отличается от A.

Более формально, зафиксируем некоторое главное универсальное перечислимое множество W (и тем самым нумерацию перечислимых множеств: число n мы считаем номером множества W_n). Будем говорить, что множество A является *эффективно неперечислимым*, если существует такая всюду определённая вычислимая функция f, что $f(z) \in A \triangle W_z$ при всех z. (Здесь $\triangle$ означает симметрическую разность; другими словами, $f(z)$ является точкой, где A отличается от W_z.)

Заметим, что это свойство не зависит от выбора главного универсального множества, так как от номеров относительно одного такого множества можно эффективно переходить к номерам относительно другого.

Свойство эффективной неперечислимости допускает простую характеризацию в терминах m-сводимости. Начнём с такого простого наблюдения.

Теорема 34. Если $A \leqslant_m B$ и A эффективно неперечислимо, то и B эффективно неперечислимо.

$\triangleleft$ Эта теорема является «эффективным вариантом» теоремы 31, часть (б). То же самое можно сказать и о её доказательстве. Пусть мы хотим найти точку, в которой B отличается от некоторого перечислимого множества X. Рассмотрим функцию f, которая m-сводит A к B. Прообраз $f^{-1}(X)$ перечислимого множества X при вычислимом отображении будет перечислим, поэтому можно найти точку m, в которой он отличается от A. Тогда B отличается от X в точке $f(m)$.

Чтобы сделать это рассуждение точным, нам нужно лишь доказать, что номер перечислимого множества $f^{-1}(X)$ может быть эффективно получен по номеру перечислимого множества X. Для этого мы должны воспользоваться тем, что нумерация является главной — схема тут та же, что и при вычислении номера композиции двух вычислимых функций, заданных своими номерами (теорема 16). Проведём это рассуждение подробно.

Рассмотрим перечислимое множество

$$V = \{\langle x, y\rangle \mid \langle x, f(y)\rangle \in W\}$$

(оно перечислимо, поскольку является прообразом перечислимого множества W при вычислимом отображении $\langle x, y\rangle \mapsto \langle x, f(y)\rangle$). Легко видеть, что $V_n = f^{-1}(W_n)$.

Так как множество W является главным универсальным множеством, то существует вычислимая всюду определённая функция s, для которой $W_{s(n)} = V_n = f^{-1}(W_n)$ при всех n. Другими словами, функция s по W-номеру любого перечислимого множества даёт W-номер его прообраза при отображении f, что и требовалось. $\triangleright$

Теорема 35. Существуют перечислимые множества с эффективно неперечислимыми дополнениями.

$\triangleleft$ Вновь рассмотрим диагональное множество $D = \{n \mid \langle n, n\rangle \in W\}$. Его дополнение будет эффективно неперечислимым. В самом деле, множества W_n и D одинаково себя ведут в точке n, поэтому W_n отличается от дополнения к D в этой точке. Таким образом, дополнение к D эффективно неперечислимо, причём в качестве функции f из определения эффективной неперечислимости можно взять тождественную функцию. $\triangleright$

Из двух предыдущих теорем очевидно следует такое утверждение:

Теорема 36. Всякое m-полное перечислимое множество имеет эффективно неперечислимое дополнение.

На самом деле верно и обратное. Чтобы убедиться в этом, докажем такой факт:

Теорема 37. Пусть K — перечислимое множество, а A эффективно неперечислимо. Тогда $\mathbb{N} \setminus K \leqslant_m A$ (или, что эквивалентно, $K \leqslant_m \mathbb{N} \setminus A$).

$\triangleleft$ На самом деле нам важно умение эффективно отличать A лишь от двух перечислимых множеств — от пустого и от всего натурального ряда. Отличить A от пустого множества означает указать элемент в A; отличить от всего натурального ряда означает указать элемент вне A. Именно эти две вещи используются при сведении. Более формально, рассмотрим множество $V = K \times \mathbb{N}$. Его сечения V_n либо пусты (при $n \notin K$), либо совпадают со всем натуральным рядом (при $n \in K$). Пользуясь тем, что множество W является главным, мы находим всюду определённую функцию s, для которой $W_{s(n)} = \varnothing$ при $n \notin K$ и $W_{s(n)} = \mathbb{N}$ при $n \in K$. Пусть f — функция, обеспечивающая эффективную неперечислимость множества A. Тогда $f(s(n)) \in A$ при $n \notin K$ и $f(s(n)) \notin A$ при $n \in K$.

Другими словами, композиция функций f и s сводит $\mathbb{N}\setminus K$ к множеству A, что и требовалось. $\rhd$

Отсюда очевидно вытекают такие утверждения:

Теорема 38. Перечислимое множество является m-полным тогда и только тогда, когда его дополнение эффективно неперечислимо.

Теорема 39. Множество эффективно неперечислимо тогда и только тогда, когда к нему m-сводится дополнение некоторого (вариант: любого) m-полного множества.

Отметим, что не всякое неперечислимое множество эффективно неперечислимо. Это видно, например, из такого факта:

Теорема 40. Любое эффективно неперечислимое множество содержит бесконечное перечислимое подмножество (т. е. не является иммунным).

$\lhd$ В самом деле, пусть множество A эффективно неперечислимо. Тогда отличим его от пустого множества — то есть найдём в нём элемент. После этого отличим его от одноэлементного множества, которое состоит из этого элемента — и найдём другой элемент. Действуя так, мы может алгоритмически найти сколь угодно много различных элементов.

В этом рассуждении мы использовали такой факт: по конечному множеству, заданному списком его элементов, можно получить его номер (точнее, один из номеров) в главной нумерации перечислимых множеств. Почему это так? Пусть фиксирована некоторая вычислимая нумерация конечных множеств. Будем обозначать n-ое конечное множество в этой нумерации через D_n. Тогда D_n будет n-ым сечением перечислимого (и даже разрешимого) множества

$$D = \{\langle n, x\rangle \mid x \in D_n\}.$$

Остаётся воспользоваться определением главной нумерации перечислимых множеств. $\rhd$

Простые множества, которые, как мы знаем, существуют (теорема 14), являются примерами перечислимых множеств, не являющихся m-полными. Именно так и возникло понятие простого множества: Пост искал пример перечислимого неразрешимого множества, которое не было бы m-полным.

6.4. Изоморфизм m-полных множеств

В этом разделе мы докажем, что все m-полные множества «устроены одинаково» и отличаются друг от друга только вычислимой перестановкой.

Теорема 41. Пусть A и B — m-полные перечислимые множества. Тогда существует вычислимая перестановка (вычислимое взаимно однозначное соответствие) $f\colon \mathbb{N} \to \mathbb{N}$, при которой A переходит в B, то есть $x \in A \Leftrightarrow f(x) \in B$ при всех x.

$\lhd$ Мы будем использовать тот же приём, что и в приведённом выше доказательстве теоремы Роджерса об изоморфизме главных нумераций (см. с. 39). Именно, мы для начала докажем такую лемму:

Лемма. Пусть A — m-полное перечислимое множество. Тогда существует способ по любому натуральному числу n алгоритмически указать сколько угодно других натуральных чисел, которые принадлежат или не принадлежат A одновременно с n.

Доказательство леммы. Как и раньше, у нас будут два способа получать новые числа, которые ведут себя по отношению к A так же, как и исходное. Один из них будет гарантированно давать новое число, если $x \in A$, другой — если $x \notin A$. При этом мы можем применять оба, не зная, какой из случаев имеет место на самом деле (и можем так этого и не узнать).

Первый способ состоит в следующем. Пусть P — перечислимое неразрешимое множество. Рассмотрим перечислимое множество пар $A \times P$. Оно сводится к A, так как A является m-полным. (Вообще-то в определении сводимости шла речь о множествах натуральных чисел, а не пар, но, как всегда, это не играет роли — пары можно вычислимо нумеровать.) Другими словами, существует вычислимая всюду определённая функция f двух натуральных аргументов с таким свойством:

$$f(n,m) \in A \Leftrightarrow (n \in A) \text{ и } (m \in P).$$

В частности, при $m \in P$ числа n и $f(n,m)$ одновременно принадлежат или не принадлежат A. Поэтому, расположив P в вычислимую последовательность $p(0), p(1), \ldots$, мы можем вычислять числа $f(n,p(0)), f(n,p(1)), \ldots$ и получать новые числа, которые принадлежат или не принадлежат A одновременно с n.

Пусть $n \in A$. Покажем, что множество X получаемых таким образом чисел (все они в этом случае тоже принадлежат A) будет бесконечно. В самом деле, $f(n,m) \in X$ при $m \in P$ (по построению X) и $f(n,m) \notin X$ при $m \notin P$ (поскольку в этом случае $f(n,m) \notin A$, а $X \subset A$). Таким образом, функция $m \mapsto f(n,m)$ сводит неразрешимое множество P к множеству X, так что X неразрешимо и потому бесконечно.

Теперь опишем другой способ, который гарантирует успех, если $n \notin A$. Возьмём два перечислимых неотделимых множества P и Q. Рассмотрим перечислимое множество пар $(A \times P) \cup (\mathbb{N} \times Q)$. Пусть функция f сводит его к A. Это означает, что $f(n,m) \in A$ тогда и только тогда, когда ($n \in A$ и $m \in P$) или $m \in Q$. Как и прежде, при $m \in P$ числа n и $f(n,m)$ одновременно принадлежат или не принадлежат A, так что мы можем снова рассмотреть последовательность $f(m,p(0)), f(n,p(1)), \ldots$; осталось лишь показать, что (если $n \notin A$) в этой последовательности бесконечно много различных членов.

Пусть это не так и множество X всех членов этой последовательности конечно. По нашему предположению X не пересекается с A. Заметим, что $f(n,m) \in X$ при $m \in P$ (по построению) и $f(n,m) \notin X$ при $m \in Q$ (так как в этом случае $\langle n,m \rangle$ принадлежит нашему перечислимому множеству пар и $f(n,m)$ принадлежит A). Таким образом, прообраз множества X при отображении $m \mapsto f(n,m)$ отделяет P от Q. Но этот прообраз разрешим (X разрешимо, ибо конечно, а указанное отображение всюду определено и вычислимо). А по нашему предположению множества P и Q нельзя отделить разрешимым множеством.

Итак, мы привели два способа получать новые элементы, которые принадлежат или не принадлежат A одновременно с исходным. Применяя их параллельно, мы наверняка добьёмся результата. Лемма доказана.

Пусть теперь A и B — два m-полных перечислимых множества. Докажем, что они отличаются лишь вычислимой перестановкой натурального ряда. Будем строить эту перестановку по шагам. На k-ом шаге мы имеем взаимно однозначное соответствие

$$a_1 \leftrightarrow b_1, a_2 \leftrightarrow b_2, \ldots, a_k \leftrightarrow b_k,$$

при котором $a_i \in A \Leftrightarrow b_i \in B$ при всех i. На чётных шагах мы берём минимальное число, не входящее в левую часть этого соответствия. Используя факт m-сводимости A к B, мы находим ему компаньона. При этом доказанная нами лемма позволяет выбрать компаньона, не встречающегося среди уже имеющихся справа элементов. На нечётных шагах мы делаем то же самое, только справа налево.

В пределе этот процесс даёт искомую вычислимую перестановку, связывающую A и B. $\triangleright$

С точки зрения теории алгоритмов два множества, отличающиеся лишь вычислимой перестановкой, обладают одинаковыми свойствами. Поэтому доказанная теорема показывает, что по существу имеется лишь одно m-полное перечислимое множество (или, что то же, лишь одно перечислимое множество с эффективно неперечислимым дополнением).

6.5. Продуктивные множества

В этом разделе мы используем теорему о неподвижной точке для получения такого (неожиданного на первый взгляд) результата: определение эффективной неперечислимости множества A не изменится, если мы ограничимся лишь (перечислимыми) подмножествами множества A.

Зафиксируем некоторую главную нумерацию перечислимых множеств (множество с номером n мы обозначаем W_n). Говорят, что множество A является *продуктивным*, если существует вычислимая (не обязательно всюду определённая) функция f с таким свойством: она применима к любому номеру n любого подмножества W_n множества A и даёт элемент в их разности:

$$W_n \subset A \Rightarrow f(n) \in A \setminus W_n.$$

48. Докажите, что продуктивное множество не может быть иммунным.

Ясно, что требования в определении продуктивности — лишь часть требований из определения эффективно неперечислимого множества, так что любое эффективно неперечислимое множество продуктивно. Удивительным образом оказывается, что верно и обратное.

Теорема 42. Пусть A — продуктивное множество, K — произвольное перечислимое множество. Тогда дополнение к K m-сводится к A.

(Из этого следует, что A является эффективно неперечислимым, см. выше.)

$\lhd$ Пусть f — функция, которая существует по определению продуктивности (и даёт элемент вне подмножества с указанным номером).

Мы построим всюду определённую вычислимую функцию s с такими свойствами:

- $x \notin K \Rightarrow W_{s(x)} = \varnothing$;
- $x \in K \Rightarrow W_{s(x)} = \{f(s(x))\}$.

(второе свойство подразумевает, что $f(s(x))$ определено при $x \in K$). Прежде чем делать это с помощью теоремы о неподвижной точке, заметим, что в первом случае $f(s(x))$ определено и принадлежит A: поскольку множество с номером $s(x)$ пусто и является подмножеством A, число $f(s(x))$ должно быть элементом A. Напротив, во втором случае $f(s(x))$ не принадлежит A. В самом деле, если бы это было не так, то множество $W_{s(x)}$ было бы подмножеством A, и потому число $f(s(x))$ должно было бы быть элементом A, не входящим в это подмножество — а оно входит.

Поэтому если нам удастся построить такую функцию s, то функция $x \mapsto f(s(x))$ будет m-сводить дополнение множества K к множеству A, как мы и обещали. Как же её строить?

Если бы во втором свойстве (для $x \in K$) стояло не $f(s(x))$, а, скажем, просто $f(x)$, никакой проблемы бы не было. Как обычно, мы рассмотрели бы перечислимое множество пар

$$V = \{\langle x, y\rangle \mid x \in K \text{ и } y = f(x)\};$$

сечения этого множества имели бы требуемый вид и осталось бы только воспользоваться тем, что нумерация — главная. Но в нашем случае, когда в правой части второго свойства стоит $f(s(x))$, так просто поступить нельзя: как в истории о курице и яйце, для построения V нам надо иметь $s(x)$, а для построения $s(x)$ надо иметь V.

Именно такого рода трудности позволяет преодолевать теорема о неподвижной точке. Построим всюду определённую вычислимую функцию двух аргументов h с такими свойствами:

- $x \notin K \Rightarrow W_{h(x,t)} = \varnothing$;
- $x \in K \Rightarrow W_{h(x,t)} = \{f(t)\}$.

(Подобные вещи мы делали многократно — последний раз в предыдущем абзаце. Отметим, что $f(t)$ может быть и не определено, тогда под $\{f(t)\}$ мы понимаем пустое множество.) По теореме о неподвижной точке (для перечислимых множеств) при каждом x функция $t \mapsto h(x,t)$ имеет неподвижную точку, и, как мы говорили в разделе о неподвижной точке с параметром, эту неподвижную точку можно выбрать вычислимо зависящей от x. Таким образом, существует всюду определённая вычислимая функция s, для которой

$$W_{s(x)} = W_{h(x,s(x))}$$

при всех x. Это равенство можно продолжить:

$$W_{s(x)} = W_{h(x,s(x))} = \begin{cases} \varnothing, \text{ если } x \notin K, \\ \{f(s(x))\}, \text{ если } x \in K \end{cases}$$

— это ровно то, чего мы и хотели. Заметим, что значение $f(s(x))$ определено при всех x (иначе $W_{s(x)} = \varnothing$, и $f(s(x))$ должно быть определено). Тем самым, теорема о неподвижной точке позволяет отыскать взаимно согласованные яйцо и курицу и завершает доказательство. $\triangleright$

Перечислимые множества, дополнения которых продуктивны, называются *креативными* (creative; иногда это слово переводят как «творческие»). Название объясняется так: это множество (точнее, его дополнение) более изобретательно, чем любой алгоритмический процесс: если кто-то предлагает способ порождать некоторые элементы из дополнения, то в ответ можно указать элемент дополнения, который нельзя получить таким способом.

Как мы видим, творческие множества, перечислимые множества с эффективно неперечислимым дополнением и m-полные множества — один и тот же класс, и любые два множества из этого класса в некотором смысле изоморфны (отличаются лишь вычислимой перестановкой).

Если множество продуктивно, то можно порождать его элементы следующим индуктивным процессом. На первом шаге имеется пустое множество. Применив к нему продуктивную функцию (т. е. функцию, существующую по определению продуктивного множества), мы получим некоторый элемент. Он образует одноэлементное подмножество. Применив к этому подмножеству продуктивную функцию, получим другой элемент. К полученному двухэлементному подмножеству можно снова применить продуктивную функцию и так далее. Получится бесконечная вычислимая последовательность элементов продуктивного множества. (Это мы уже делали, когда доказывали, что эффективно неперечислимое множество содержит бесконечное перечислимое подмножество.) Но этот индуктивный процесс можно «трансфинитно» продолжить, по крайней мере ещё немного: имея перечислимое подмножество нашего продуктивного множества (множество членов последовательности), можно найти ещё один элемент продуктивного множества (так сказать, элемент номер ω). Добавим его к последовательности, снова применим продуктивную функцию, получится $(\omega+1)$-ый элемент и так далее, затем получится новая последовательность, $(\omega 2)$-ый элемент, $(\omega 3)$-ый, ..., ω^2-элемент и т. д.

Но, конечно, получить таким образом алгоритм, перечисляющий продуктивное (и потому неперечислимое) множество, не удастся.

49. Не используя теорему о неподвижной точке (и теорему 42), покажите, что для всякого продуктивного множества A существует *всюду определённая* вычислимая функция f, для которой $W_n \subset A$ влечёт $f(n) \in A \setminus W_n$. (Указание: чередуйте W_n с пустым множеством, как это делается при доказательстве леммы к теореме 41.)

6.6. Пары неотделимых множеств

В этом разделе мы сформулируем некоторые результаты, касающиеся пар непересекающихся перечислимых множеств. Эти результаты параллельны только что доказанным нами теоремам о m-полноте, продуктивности, эффективной неперечислимости и об изоморфизме m-полных множеств.

Пусть A и B — два непересекающихся множества (натуральных чисел). Напомним, что они называются неотделимыми, если не существует разрешимого множества, содержащего одно из них и не пересекающегося с другим. Это определение можно переформулировать так: если W_x и W_y — два непересекающихся перечислимых множества, содержащие A и B соответственно, то объединение $W_x \cup W_y$ содержит не все натуральные числа. (Нам будет удобно обозначать перечислимые множества через W_x и W_y, считая, что W — главное универсальное множество.)

Теперь ясно, как можно сформулировать эффективный вариант этого определения. Будем говорить, что непересекающиеся множества A и B *эффективно неотделимы*, если существует вычислимая функция h с таким свойством: если $A \subset W_x$, $B \subset W_y$ и $W_x \cap W_y = \varnothing$, то $h(x, y)$ определено и $h(x, y) \notin W_x \cup W_y$.

Определение неотделимости можно сформулировать чуть-чуть иначе: не существует вычислимой функции φ_n, которая была бы всюду определённой, во всех точках множества A равнялась бы нулю, а во всех точках множества B — единице. (Будем считать, что φ — главная универсальная функция.) Соответственно изменится и эффективный вариант: множества A и B *сильно эффективно неотделимы*, если существует всюду определённая вычислимая функция h, которая по любому n указывает точку $h(n)$, в которой функция φ_n «ошибается». Ошибка возможна трёх видов: либо $\varphi_n(h(n))$ не определено, либо $h(n) \in A$, но $\varphi_n(h(n))$ не равно нулю, либо $h(n) \in B$, но $\varphi_n(h(n))$ не равно единице.

50. Покажите, что из сильной эффективной неотделимости вытекает эффективная неотделимость (что оправдывает используемую нами терминологию).

Обратное утверждение также верно, но доказывается несколько сложнее, и мы к нему ещё вернёмся.

Существуют ли сильно эффективно неотделимые перечислимые множества? Легко понять, что стандартная диагональная конструкция даёт пару таких множеств, а именно множества $\{x \mid \varphi_x(x) = 1\}$ и $\{x \mid \varphi_x(x) = 0\}$, для которых в качестве функции h можно взять тождественную функцию.

51. Проверьте это.

Продолжая нашу аналогию (между множествами и парами), определим понятие m-сводимости для пар. Здесь тоже будет два варианта. Пусть $\langle A, B\rangle$ и $\langle C, D\rangle$ — две пары непересекающихся перечислимых множеств (A не пересекается с B, а C — с D). Будем говорить, что вычислимая всюду определённая функция f *m-сводит* $\langle A, B\rangle$ *к* $\langle C, D\rangle$, если $f(A) \subset C$ и $f(B) \subset D$.

52. (**а**) Покажите, что если f сводит $\langle A, B\rangle$ к $\langle C, D\rangle$ и C отделимо от D разрешимым множеством, то и A отделимо от B разрешимым множеством. (**б**) Покажите, что если f сводит $\langle A, B\rangle$ к $\langle C, D\rangle$ и пара $\langle A, B\rangle$ эффективно неотделима, то и пара $\langle C, D\rangle$ эффективно неотделима. (**в**) Покажите, что если f сводит $\langle A, B\rangle$ к $\langle C, D\rangle$ и пара $\langle A, B\rangle$ сильно эффективно неотделима, то и пара $\langle C, D\rangle$ сильно эффективно неотделима.

Определение сводимости можно усилить, потребовав дополнительно, чтобы при $x \notin A \cup B$ выполнялось $f(x) \notin$ $\notin C \cup D$ (другими словами, f должна сводить A к C и одновременно B к D). В этом случае мы будем говорить, что f *сильно сводит* пару $\langle A, B\rangle$ к паре $\langle C, D\rangle$.

Теперь мы можем определить *m-полноту* и *сильную m-полноту* для пары непересекающихся перечислимых множеств, требуя m-сводимости (соответственно сильной m-сводимости) любой такой пары к данной.

53. Покажите, что если пара является сильно эффективно неотделимой, то она является сильно m-полной. (Указание. Пусть пара $\langle A, B\rangle$ сильно эффективно неотделима, а $\langle K, L\rangle$ — любая пара непересекающихся перечислимых множеств. По любому натуральному числу x можно построить вычислимую функцию ψ_x с таким свойством: если $x \in K$, то ψ_x всюду определена и отличается от единицы лишь в конечном числе точек, причём все эти точки принадлежат A; если $x \in L$, то ψ_x всюду определена и отличается от нуля лишь в конечном числе точек, причём все эти точки принадлежат B; если $x \notin K \cup L$, то ψ_x равна нулю на A и единице на B. Чтобы построить такую функцию, перечисляем K и L; пока x не обнаружилось в одном из этих множеств, добавляем в график ψ_x пары вида $\langle a, 0\rangle$ и $\langle b, 1\rangle$; когда x обнаруживается, перестраиваемся. Далее остаётся воспользоваться свойствами главной нумерации φ и сильной эффективной неотделимостью A и B.)

54. Покажите, что всякая m-полная пара является сильно эффективно неотделимой. (Указание: сильно эффективно неотделимая пара существует и к ней сводится.)

Из сформулированных в качестве задач утверждений вытекает, что свойства полноты, сильной m-полноты и сильной эффективной неотделимости эквивалентны. Можно доказать, что и кажущееся более слабым свойство эффективной неотделимости эквивалентно им. Рассуждение при этом аналогично доказательству теоремы 42 о том, что всякое креативное множество является m-полным. Заметим, что разница между эффективной неотделимостью и сильной эффективной неотделимостью примерно такая же, как между продуктивностью и эффективной неперечислимостью.

55. Пусть $\langle A, B\rangle$ — эффективно неотделимая пара непересекающихся множеств. Покажите, что она является сильно m-полной. (Указание. Пусть K и L — произвольные непересекающиеся перечислимые множества. Пусть h — функция из определения эффективной неотделимости (множеств A и B). С помощью теоремы о неподвижной точке постройте всюду определённые вычислимые функции $x(n)$ и $y(n)$ с такими свойствами: (1) если $n \in K$, то $W_{x(n)} = A$, $W_{y(n)} = B \cup \cup\{h(x(n), y(n))\}$; (2) если $n \in L$, то $W_{x(n)} = A \cup \{h(x(n), y(n))\}$, $W_{y(n)} = B$; (3) если $n \notin K \cup L$, то $W_{x(n)} = A$, $W_{y(n)} = B$. Выведите отсюда, что при $n \in K$ значение $h(x(n), y(n))$ определено и принадлежит A, при $n \in L$ значение $h(x(n), y(n))$ определено и принадлежит B, а при $n \notin K \cup L$ значение $h(x(n), y(n))$ определено и лежит вне $A \cup B$.)

Итак, все четыре сформулированных свойства эквивалентны. Продолжая нашу аналогию, можно доказать изоморфность любых двух пар эффективно неотделимых множеств, для чего предварительно научиться получать сколь угодно много чисел, «эквивалентных» данному с точки зрения пары эффективно неотделимых множеств.

Более точно, пусть имеются непересекающиеся множества A и B. Назовём два числа $\langle A, B\rangle$-эквивалентными в любом из следующих трёх случаев: оба они принадлежат A, оба они принадлежат B или оба они не принадлежат $A \cup B$. (Таким образом, есть три класса эквивалентности — множество A, множество B и остаток.)

56. Пусть $\langle A, B\rangle$ — сильно m-полная пара перечислимых множеств. Покажите, что по любому числу k можно алгоритмически получать сколь угодно много различных чисел, которые будут $\langle A, B\rangle$-эквивалентны k. (Указание: действуйте по аналогии с доказательствами теоремы 22 и леммы к теореме 41.)

57. Пусть $\langle A_1, B_1\rangle$ и $\langle A_2, B_2\rangle$ — две сильно m-полные пары перечислимых множеств. Тогда они вычислимо изоморфны в следующем смысле: существует вычислимая перестановка (биекция) $i\colon \mathbb{N} \to \mathbb{N}$, при которой $i(A_1) = B_1$ и $i(A_2) = B_2$. (Указание: действуйте по аналогии с доказательствами теорем 23 и 41.)

7. Вычисления с оракулом

7.1. Машины с оракулом

Если множество B m-сводится к разрешимому множеству A, то и B разрешимо. Более того, если даже A и неразрешимо, но у нас есть доступ к «оракулу» для A, который отвечает на вопросы о принадлежности чисел множеству A, то мы можем с его помощью отвечать на вопросы о принадлежности чисел множеству B. В самом деле, если f — сводящая функция и если мы хотим узнать, принадлежит ли некоторое число x множеству B, достаточно спросить у оракула, принадлежит ли $f(x)$ множеству A.

Легко видеть, что m-сводимость использует возможности оракула довольно ограниченным образом: во-первых, оракулу задаётся только один вопрос, во-вторых, ответ на этот вопрос и считается ответом на исходный вопрос о принадлежности числа x множеству A. Вот пример, не укладывающийся в такую схему: имея оракул для множества A, мы можем отвечать на вопросы о принадлежности чисел множеству $B = \mathbb{N} \setminus A$. Здесь вопрос по-прежнему один, но ответ на него заменяется на противоположный. Другой пример: имея оракул для множества A, можно отвечать на вопросы о принадлежности пары натуральных чисел множеству $B = A \times A$. (Здесь оракулу надо задать уже два вопроса.)

Поэтому естественно желание отказаться от этих ограничений и дать общее определение сводимости множества B к множеству A. Наиболее общее и естественное определение таково: B сводится к A, если существует алгоритм, который разрешает множество B при условии, что ему предоставлен доступ к оракулу, отвечающему на вопросы про множество A. В более программистских терминах: есть алгоритм, содержащий вызовы внешней функции `a(x:integer):boolean` (не описанной внутри алгоритма); этот алгоритм разрешает множество B, если вызовы `a(x)` возвращают правильные ответы про множество A.

Этот вид сводимости называется *сводимостью по Тьюрингу*, или T-сводимостью. Обозначение: $B \leqslant_T A$ означает, что B сводится по Тьюрингу к A. Вот несколько простых фактов про T-сводимость:

Теорема 43. (а) Если $B \leqslant_m A$, то $B \leqslant_T A$. (б) $A \leqslant_T \leqslant_T \mathbb{N} \setminus A$ при любом A. (в) Если $A \leqslant_T B$ и $B \leqslant_T C$, то $A \leqslant_T C$. (г) Если $A \leqslant_T B$ и B разрешимо, то A разрешимо.

$\lhd$ Все эти утверждения почти очевидны — поясним, например, утверждение (в). Пусть у нас есть алгоритм для A, включающий вызовы внешней разрешающей процедуры для B, а также алгоритм для B, включающий вызовы внешней процедуры для C. Тогда можно заменить вызовы внешней B-процедуры на этот второй алгоритм и получится разрешающий алгоритм для A, использующий вызовы внешней процедуры для C. $\rhd$

Заметим, что (в отличие от m-сводимости) неперечислимое множество вполне может T-сводиться к перечислимому. Например, дополнение перечислимого неразрешимого множества K сводится к самомý множеству K.

Можно говорить не только о сводимости к множеству A, но и вообще об алгоритмах, имеющих доступ к оракулу для множества A. Пусть такой алгоритм вычисляет некоторую функцию f. Это означает, напомним, что если $f(x)$ определено, то на входе x алгоритм останавливается и даёт ответ $f(x)$, а если $f(x)$ не определено, то не останавливается. (Предполагается, естественно, что оракул «не зависает» и выдаёт ответы, притом правильные, на все заданные ему вопросы.) В этом случае говорят, что (частичная) функция f вычислима относительно множества A.

В нашем определении сводимости вызываемая внешняя функция принимала только два значения («да» и «нет»). Такое ограничение вовсе не обязательно. Пусть $\alpha\colon \mathbb{N} \to \mathbb{N}$ — произвольная всюду определённая функция. Тогда можно говорить о функциях, *вычислимых относительно* α; вычисляющие их алгоритмы включают в себя вызовы функции α. Однако это обобщение не является существенным:

Теорема 44. Частичная функция f вычислима относительно всюду определённой функции α тогда и только тогда, когда она вычислима относительно множества, являющегося графиком функции α, то есть относительно множества $\{\langle n, \alpha(n)\rangle \mid n \in \mathbb{N}\}$.

$\triangleleft$ В самом деле, если мы можем вызывать функцию α, то можем и отвечать на вопросы о принадлежности произвольной пары графику функции α. Напротив, если мы можем разрешать график α, то можем найти $\alpha(x)$ для данного x, задавая по очереди вопросы о принадлежности графику пар $\langle x, 0\rangle, \langle x, 1\rangle, \ldots$, пока не получим положительный ответ. $\triangleright$

Определяя вычислимость относительно функции α, мы предполагали, что α всюду определена. Это ограничение принципиально: для не всюду определённых функций механизм обращения к ним (как к внешним процедурам) требует уточнений. Допустим, мы вызвали $\alpha(x)$, а оказалось, что функция α не определена на x. Означает ли это, что алгоритм в этом месте «зависает» и уже не может выдать результат? Или мы можем параллельно развернуть какие-то вычисления и в каких-то случаях выдать результат, не дожидаясь ответа от $\alpha(x)$? Можем ли мы параллельно запросить несколько значений функции α? Например, следует ли считать функцию $f(x)$, заданную формулой

$$f(x) = \begin{cases} 0, \text{ если } \alpha(2x) \text{ или } \alpha(2x+1) \text{ определено,} \\ \text{не определено в противном случае} \end{cases}$$

вычислимой относительно α? Короче говоря, в отличие от случая всюду определённых функций, тут есть разные (и притом не эквивалентные) варианты определений, и всегда надо уточнять, какое именно понятие имеется в виду. Поэтому мы, говоря о вычислимости относительно некоторой функции α, предполагаем, что функция α всюду определена.

58. Пусть есть два различных множества X и Y. Будем рассматривать программы, имеющие доступ к двум оракулам для X и для Y, и функции, которые можно вычислить с помощью таких программ. Покажите, что это определение не даёт ничего существенно нового, указав такое множество Z, что X-Y-вычислимость совпадает с Z-вычислимостью.

7.2. Относительная вычислимость: эквивалентное описание

Сейчас мы дадим эквивалентное определение вычислимости функции относительно α, не апеллирующее к программам с вызовом оракула.

Мы называли образцом функцию с натуральными аргументами и значениями, определённую на конечном подмножестве натурального ряда. Такой образец задаётся списком пар $\langle$аргумент, значение$\rangle$; образцы можно вычислимо пронумеровать, после чего не различать образец и его номер и говорить о разрешимом множестве образцов, перечислимом множестве образцов и т. д.

Два образца называются *совместными*, если объединение их графиков есть по-прежнему график функции, то есть если нет такой точки, в которой оба они были бы определены и принимали разные значения.

Пусть имеется множество M троек вида $\langle x, y, t\rangle$, где x и y — натуральные числа, а t — образец. Будем говорить, что две тройки $\langle x_1, y_1, t_1\rangle$ и $\langle x_2, y_2, t_2\rangle$ *противоречат* друг другу, если $x_1 = x_2$, $y_1 \ne y_2$, а образцы t_1 и t_2 совместны. Множество M будем называть *корректным*, если в нём нет противоречащих друг другу троек.

Пусть M — корректное множество, а α — некоторая функция. Отберём в M все тройки вида $\langle x, y, t\rangle$, для которых t является частью α (график t является подмножеством графика α). Входящие в них образцы совместны, поэтому (в силу корректности) среди отобранных троек нет двух, у которых первые члены равны, а вторые — нет. Значит, отбросив третьи компоненты в отобранных тройках, мы получим график некоторой функции (вообще говоря, частичной). Будем обозначать эту функцию $M[\alpha]$.

Теорема 45. Частичная функция $f\colon \mathbb{N} \to \mathbb{N}$ вычислима относительно всюду определённой функции $\alpha\colon \mathbb{N} \to \mathbb{N}$ тогда и только тогда, когда существует перечислимое корректное множество троек M, для которого $f = M[\alpha]$.

$\triangleleft$ Пусть имеется программа p, вычисляющая f и включающая в себя обращения к внешней процедуре α. Будем для всех натуральных n моделировать работу этой программы на входе n по всем путям, то есть предусматривая все возможные значения $\alpha(n)$ для каждого обращения к внешней процедуре. Для каждого n получается дерево вариантов — каждому обращению к внешней процедуре соответствует развилка со счётным ветвлением. На некоторых ветвях этого дерева вычисления завершаются и программа выдаёт ответ. Как только мы обнаруживаем, что на входе x возможен ответ y (на некоторой ветви), мы образуем тройку $\langle x, y, t\rangle$, где t — образец, содержащий все аргументы и значения функции α, использованные на этой ветви.

Полученное множество троек, которое мы обозначим M, будет перечислимым (описанная процедура позволяет выписывать все его элементы). В этом множестве нет противоречащих друг другу троек. В самом деле, если для одного и того же x в него вошли тройки $\langle x, y_1, t_1\rangle$ и $\langle x, y_2, t_2\rangle$ с $y_1 \neq y_2$, то они соответствуют разным путям в дереве вычислений на одном и том же входе x. Эти пути в каком-то месте разошлись, то есть на один и тот же вопрос в них были получены разные ответы. Эти ответы вошли в образцы t_1 и t_2, и потому эти образцы несовместны. Итак, множество M корректно.

Пусть α — всюду определённая функция. Присоединим её к программе p. После этого программа p вычисляет функцию f. Покажем, что $f = M[\alpha]$. В самом деле, пусть $f(x) = y$, то есть работа программы p на входе x дала ответ y. Эта работа включала в себя несколько вызовов функции α и соответствовала некоторой ветви рассмотренного выше дерева. Пусть t — образец, содержащий все заданные при этом вопросы и полученные на них ответы. Тогда t является частью α. Кроме того, тройка $\langle x, y, t\rangle$ входит в множество M. Следовательно, $M[\alpha](x)$ определено и равно y.

Напротив, если $M[\alpha](x) = y$, то существует тройка $\langle x, y, t\rangle \in M$, для которой t является частью α. Эта тройка соответствует некоторой ветви дерева вычислений. Поскольку t является частью α, присоединение к программе p внешней процедуры α приведёт к тому, что вычисления пойдут именно по этому пути, и программа даст ответ y.

Итак, для любой программы p мы построили корректное множество M, которое задаёт ту же функцию, что и программа p, и первая половина утверждения теоремы доказана.

Чтобы доказать вторую половину, предположим, что имеется корректное множество M, и построим эквивалентную ему программу p. Эта программа будет (после присоединения к ней оракула, вычисляющего α) вычислять функцию $M[\alpha]$. Программа p действует так: получив вход x, она перечисляет множество M и отбирает в нём тройки, первым членом которых является x. Для каждой такой тройки $\langle x, y, t\rangle$, вызывая внешнюю процедуру (задавая вопросы оракулу) мы выясняем, является ли t частью функции α. Если является, то вычисление заканчивается и выдаётся ответ y, если нет, перечисление множества M продолжается.

Очевидно, что построенная программа p вычисляет функцию $M[\alpha]$. $\triangleright$

59. Предположим, что мы провели это построение в обе стороны: сначала по корректному множеству M построили некоторую программу, как это описано во второй половине доказательства, а затем по программе построили некоторое корректное множество M'. Может ли M' отличаться от M?

7.3. Релятивизация

Пусть фиксирована некоторая всюду определённая функция α. Тогда вся теория вычислимых функций может быть, как говорят, «релятивизована» относительно α, если во всех определениях и формулировках заменить вычислимые функции на функции, вычислимые относительно α (которые для краткости называют также *α-вычислимыми*). При этом все сформулированные выше результаты остаются в силе, и доказательства остаются почти такими же.

В частности, можно определить понятие *перечислимого относительно* α (или α-*перечислимого*) множества любым из эквивалентных способов: как область определения α-вычислимой функции, как множество значений α-вычислимой функции, как проекцию α-разрешимого (разрешимого относительно α) множества и т. д. Можно указать и более прямое описание класса α-перечислимых множеств.

Пусть E — произвольное множество пар вида $\langle x, t\rangle$, где x — число, а t — образец. Пусть α — некоторая всюду определённая функция. Отберём в множестве E те пары, у которых вторые члены являются частью α; первые члены таких пар образуют множество, которое мы обозначим $E[\![\alpha]\!]$.

Теорема 46. Множество X является α-перечислимым тогда и только тогда, когда $X = E[\![\alpha]\!]$ для некоторого перечислимого множества E. (Заметим, что в этом случае не требуется никакого специального условия типа корректности.)

$\triangleleft$ Пусть X есть область определения вычислимой относительно α функции f. Тогда $f = M[\alpha]$ для некоторого перечислимого корректного множества M. Оставим от всех троек в M только первый и третий члены; получится некоторое перечислимое множество E. Легко проверить, что $E[\![\alpha]\!]$ будет областью определения функции $M[\alpha] = f$, так что $E[\![\alpha]\!] = X$.

Напротив, пусть $X = E[\![\alpha]\!]$ для некоторого α. Тогда рассмотрим множество M, которое получится, если в середину каждой пары из E добавить число 0. Ясно, что множество M будет корректным и что $M[\alpha]$ будет функцией, определённой на $X = E[\![\alpha]\!]$ и принимающей только нулевые значения. $\triangleright$

В обычной (нерелятивизованной) теории алгоритмов, важную роль играет теорема об универсальной функции. Она остаётся верной и после релятивизации:

Теорема 47. Пусть α — всюду определённая функция. Существует вычислимая относительно α функция двух аргументов, являющаяся универсальной для класса вычислимых относительно α функций одного аргумента.

$\triangleleft$ Как и в других случаях, можно почти без изменений воспроизвести доказательство соответствующей нерелятивизованной теоремы. Фиксируем какой-то язык программирования (предусматривающий на этот раз вызовы внешних процедур) и перенумеруем все программы, которые включают в себя вызовы внешней процедуры α. Теперь в качестве универсальной можно взять функцию

$$U_\alpha(i,x) = (\text{результат применения } i\text{-ой программы к } x).$$

Мы использовали нижний индекс, чтобы подчеркнуть, что функция U_α зависит от α. Впрочем, текст вычисляющей её программы от α не зависит (хотя, естественно, содержит вызовы функции α).

Поучительно привести и другое доказательство, которое опирается на определение вычислимости в терминах корректных перечислимых множеств.

Рассмотрим универсальное перечислимое множество Z четвёрок вида $\langle n,x,y,t\rangle$, где n, x и y — числа, а t — образец. Говоря об универсальности, мы имеем в виду, что при различных n среди сечений Z_n содержатся все перечислимые множества троек.

Среди этих перечислимых множеств троек могут быть и корректные, и некорректные. Мы хотим принудительно корректировать некорректные сечения, не меняя корректных. Другими словами, мы хотим построить новое перечислимое множество Z' с такими свойствами: во-первых, все сечения Z' корректны; во-вторых, если сечение Z_n при некотором n было корректно, то оно не изменилось ($Z'_n = Z_n$).

Это делается просто: нужно перечислять Z, отбрасывая (не пропуская в Z') элементы, добавление которых делает некоторое сечение некорректным. Итак, мы построили перечислимое множество Z', универсальное для класса корректных перечислимых множеств.

Теперь легко указать корректное множество W, задающее универсальную α-вычислимую функцию. Именно, тройка $\langle\langle n,x\rangle,y,t\rangle$ (теперь её первым членом является пара, так как универсальная функция зависит от двух аргументов) принадлежит W, если $\langle n,x,y,t\rangle \in Z'$. Легко понять, что множество W корректно. При данной функции α это корректное множество задаёт некоторую α-вычислимую функцию U_α двух аргументов; её n-ое сечение есть $Z'_n[\alpha]$, где Z'_n — n-ое сечение множества Z'. Поэтому среди сечений функции U_α встречаются все α-вычислимые функции, что и требовалось доказать. $\triangleright$

В релятивизованной теории алгоритмов имеется, конечно, и аналог понятия главной универсальной функции: α-вычислимую функцию двух аргументов называют *главной универсальной функцией для класса α-вычислимых функций одного аргумента*, если она α-вычислима, универсальна для класса α-вычислимых функций одного аргумента и для всякой α-вычислимой функции V двух аргументов существует всюду определённая α-вычислимая функция s одного аргумента («транслятор»), для которой $V(n,x) = U(s(n),x)$ при всех n и x.

Обычное доказательство (с. 25, теорема 15) показывает, что главные универсальные функции для класса α-вычислимых функций существуют. Более того, можно заметить, что построенная при доказательстве (см. выше) функция s будет не только α-вычислимой, но и просто вычислимой (в одном из вариантов доказательства функция s имела вид $x \mapsto [n,x]$, где квадратные скобки обозначают фиксированную вычислимую нумерацию пар, а n — некоторое фиксированное число).

Удобно, говоря о номерах α-вычислимых функций, иметь в виду их номера в таких «сильно главных» нумерациях. Естественная нумерация (порядковые номера программ) является «сильно главной» нумерацией.

Говоря о релятивизованной теории алгоритмов, иногда употребляют такую метафору. Пусть A — какое-то неразрешимое множество. Может оказаться, что есть такая внеземная цивилизация, которой множество A кажется разрешимым; глядя на число x, они сразу понимают, лежит ли оно в множестве A или нет, и эта проверка — такое же элементарное действие в их программах, как у нас сравнение двух чисел. Тогда вся их теория алгоритмов будет автоматически релятивизованной относительно A, но они этого замечать не будут и потому прочтут наши рассуждения вплоть до этого раздела (не включая его) и согласятся со всеми теоремами. Более того, они могут прочесть и этот раздел о релятивизации — но то, что для них будет B-вычислимым, для нас будет A-B-вычислимым (вычислимым с двумя оракулами A и B).

Впрочем, к этой метафоре не стоит относиться слишком серьёзно.

7.4. $\mathbf{0}'$-вычисления

В этом разделе мы рассмотрим вычислимость относительно m-полного перечислимого множества. Любые два таких множества m-сводятся друг к другу, и тем более T-сводятся друг к другу. Поэтому если какая-то функция вычислима относительно одного из них, то она вычислима и относительно другого. Такие функции называют $\mathbf{0}'$-*вычислимыми*.

Вспоминая, что множество пар $\{\langle p, x\rangle \mid$ программа p завершает работу на входе $x\}$ является одним из m-полных перечислимых множеств, можно сказать, что $\mathbf{0}'$-вычислимые функции вычисляются машинами, которым придан специальный оракул, решающий проблему остановки: этому оракулу посылают программу и вход, и он отвечает, останавливается ли эта программа на этом входе или не останавливается. (При этом посылаемая на экспертизу программа — самая обычная, без обращений к оракулу.)

Ясно, что любое перечислимое множество является $\mathbf{0}'$-разрешимым, так как сводится к m-полному перечислимому множеству. (Обратное, очевидно, неверно — дополнение к перечислимому неразрешимому множеству также $\mathbf{0}'$-разрешимо, но не перечислимо.)

Имеется следующее простое описание $\mathbf{0}'$-вычислимых функций:

Теорема 48. (а) Пусть T — всюду определённая вычислимая функция двух натуральных аргументов. Перейдём к пределу по второму аргументу, рассмотрев функцию

$$t\colon x \mapsto \lim_{n\to\infty} T(x,n).$$

(Эта функция уже не обязана быть всюду определённой, так как при некоторых x указанный предел может не существовать.) Функция t будет $\mathbf{0}'$-вычислимой. (б) Всякая $\mathbf{0}'$-вычислимая функция t может быть получена указанным образом из некоторой вычислимой всюду определённой функции T.

$\lhd$ (а) Пусть T — вычислимая всюду определённая функция двух аргументов. Назовём пару $\langle x,n\rangle$ стабильной, если $T(x,n)=T(x,m)$ для данного x и для всех $m>n$. Заметим, что множество нестабильных пар перечислимо (найдя две пары $\langle x,n\rangle$ и $\langle x,m\rangle$ с $n<m$ и $T(x,n)\neq T(x,m)$, мы включаем пару $\langle x,n\rangle$ в перечисление всех нестабильных пар). Поэтому множество нестабильных пар $\mathbf{0}'$-разрешимо. Другими словами, $\mathbf{0}'$-алгоритм для любой пары может проверить, стабильна ли она.

Рассмотрим теперь следующий $\mathbf{0}'$-алгоритм вычисления предельной функции t. Получив вход x, мы рассматриваем по очереди пары $\langle x,0\rangle,\langle x,1\rangle,\ldots$ и для каждой из них проверяем, является ли она стабильной. Как только стабильная пара $\langle x,n\rangle$ будет обнаружена, значение $T(x,n)$ выдаётся в качестве результата. Очевидно, описанный $\mathbf{0}'$-алгоритм вычисляет функцию t.

(б) Докажем теперь обратное утверждение. Пусть t — частичная $\mathbf{0}'$-вычислимая функция одного аргумента. Нам надо построить вычислимую (в обычном смысле) всюду определённую функцию двух аргументов T, для которой

$$t(x)=\lim_{n\to\infty} T(x,n)$$

при всех x (и обе части этого равенства определены одновременно). Прежде всего мы сделаем себе небольшое послабление, разрешив функции T принимать также и некоторое специальное значение, которое мы будем обозначать звёздочкой. При этом $\lim\limits_{n\to\infty} T(x,n)=a$ означает, что при всех достаточно больших n значение $T(x,n)$ равно a (и, в частности, не равно $\star$).

Такое послабление на самом деле несущественно: если в последовательности, в которой есть звёздочки, каждую из них заменить на два различных подряд идущих члена (всё равно каких), то последовательность будет иметь прежний предел (или по-прежнему не иметь предела).

Теперь определим функцию T. По предположению функция t вычисляется некоторой программой p, имеющей доступ к характеристической функции некоторого перечислимого множества K. Обозначим через K_n конечное подмножество множества K, состоящее из тех его элементов, которые успели обнаружиться за n шагов перечисления множества K. Вычисляя $T(x,n)$, мы сделаем n шагов работы программы p, при этом используя вместо K его конечное приближение K_n. Если за эти n шагов программа p не даст ответа (что может быть по разным причинам — отведённое ей время может быть недостаточно, K_n может отличаться от K, да и вообще функция t на x может быть не определена), то $T(x,n) = \star$. Если же за n шагов программа ответ даст, то этот ответ и будет значением $T(x,n)$ (за одним исключением, о котором мы скажем позже).

Попробуем доказать, что $t(x) = \lim_{n\to\infty} T(x,n)$. Пусть $t(x)$ равно некоторому a. Тогда работа программы p (с правильным оракулом K) через некоторое время завершается и даёт ответ a. При этом вычислении используется лишь конечное число вопросов к оракулу. Поэтому при достаточно большом n множество K_n в этих местах уже будет совпадать с K. Увеличив n ещё, если надо (чтобы оно превзошло время работы программы p), мы можем гарантировать, что при этом n и при всех бóльших n значение $T(x,n)$ будет равно a.

Но нам надо ещё доказать, что если предел существует и равен a, то $t(x) = a$. Здесь нас ожидает трудность, состоящая в следующем. Пусть при настоящем K работа программы p не завершается. Но тем не менее может получиться так, что при каждом n наше вычисление завершится за счёт того, что множество K_n отличается от настоящего K, и даже случайно все эти вычисления дадут одинаковый ответ.

Чтобы справиться с этой трудностью, изменим определение функции T. А именно, договоримся, что если при вычислении $T(x,n)$ и $T(x,n-1)$ протоколы обращений к оракулу были разными (задавались разные вопросы или были получены разные ответы на одинаковые вопросы), то $T(x,n)=\star$. Это не портит нашего предыдущего рассуждения, поскольку там при бóльших n задаваемые вопросы и даваемые ответы такие же, как в «настоящем» вычислении. Зато теперь мы можем быть уверены, что если последовательность $T(x,0), T(x,1), \ldots$ имеет предел, то и $t(x)$ определено. В самом деле, если она имеет предел, то содержит конечное число звёздочек. Значит, при всех достаточно больших n оракулу задаются одни и те же вопросы и получаются одни и те же ответы. Значит, эти ответы правильны, так как в пределе K_n стремится к K. Поэтому настоящее вычисление также завершается (с тем же ответом). $\triangleright$

60. Приведённое в задаче 14 (с. 16) определение вычислимого действительного числа можно релятивизовать относительного любого множества A. Покажите, что число α является $\mathbf{0}'$-вычислимым тогда и только тогда, когда оно является пределом вычислимой последовательности рациональных чисел.

7.5. Несравнимые множества

Определение сводимости по Тьюрингу (напомним, что A сводится по Тьюрингу к B, если множество A разрешимо с оракулом для B) можно рассматривать как способ сравнивать задачи разрешения различных множеств «по трудности». (Если $A \leqslant_T B$, то задача разрешения множества A в некотором смысле проще, чем задача разрешения множества B.)

Возникает множество естественных вопросов, связанных с такой классификацией. Например, существует ли самая трудная в мире задача разрешения, то есть такое множество A, что $B \leqslant_T A$ для любого множества B?

Ответ, как легко понять, отрицательный: в релятивизованном относительно A мире есть свои неразрешимые множества (и даже A-перечислимые A-неразрешимые множества) — поскольку там выполнены обычные теоремы теории алгоритмов. (Можно также заметить, что поскольку различных программ счётное число, то при любом множестве A семейство всех A-разрешимых множеств счётно.)

Другой, менее тривиальный вопрос такой: любые ли два множества сравнимы? Оказывается, что нет, как показывает следующая теорема, доказанная Клини и Постом.

Теорема 49. Существуют два множества A и B, для которых $A \not\leqslant_T B$ и $B \not\leqslant_T A$. Эти множества можно взять $\mathbf{0}'$-разрешимыми.

$\lhd$ Множества A и B должны удовлетворять таким требованиям: никакая программа, к которой присоединён B-оракул, не разрешает множества A, и никакая программа, к которой присоединён A-оракул, не разрешает множества B.

Таким образом, имеется счётное число требований (поскольку есть счётное число программ). Мы будем обслуживать их по очереди, каждое по одному разу — обеспечив выполнение некоторого требования, мы уже к нему возвращаться не будем. После каждого шага будет фиксировано поведение множеств A и B на некоторых отрезках натурального ряда, гарантирующее выполнение уже рассмотренных требований. На следующем шаги эти отрезки будут больше, и так далее — в пределе получатся два множества A и B, удовлетворяющие всем требованиям. Вся конструкция будет $\mathbf{0}'$-вычислимой, так что результирующие множества будут $\mathbf{0}'$-разрешимыми.

Опишем рассуждение более подробно. Назовём *фрагментом* функцию, которая определена на некотором (конечном) начальном отрезке натурального ряда и принимает значения 0 и 1. Будем говорить, что множество A *согласовано* с фрагментом a, если характеристическая функция множества A продолжает a. Другими словами, согласованность с данным фрагментом означает определённое поведение множества на начальном отрезке натурального ряда.

Если фрагмент a_2 продолжает фрагмент a_1 (то есть определён на большем отрезке с сохранением прежних значений на меньшем), то, очевидно, согласованность с ним накладывает больше ограничений на множество.

Лемма. Пусть a и b — два фрагмента, а p — программа, содержащая вызовы внешней процедуры. Тогда существуют продолжения a' и b' этих фрагментов с таким свойством: ни для каких множеств A и B, согласованных с a' и b', программа p, имея доступ к характеристической функции для B, не будет разрешать множество A.

Доказав эту лемму, можно поочерёдно рассматривать все программы и гарантировать, что ни одна из них не разрешает A относительно B. Если при этом чередовать A и B в применении этой леммы, то одновременно можно гарантировать, что ни одна программа не разрешает B относительно A.

(Замечание. Очевидно, можно предполагать, что a' и b' длиннее a и b — их всегда можно искусственно удлинить. Тогда в пределе мы получим бесконечные последовательности, являющиеся характеристическими функциями двух искомых множеств. Впрочем, это не обязательно: если в пределе вдруг получатся конечные фрагменты — возьмём любые множества, с ними согласованные.)

Итак, для построения множеств A и B осталось доказать лемму. (К вопросу о $\mathbf{0}'$-вычислимости мы ещё вернёмся.)

В формулировку леммы множества A и B входят несимметрично, поэтому и рассуждение будет несимметричное. Фиксируем некоторое число x, которое не входит в область определения фрагмента a, и зададим себе вопрос: существует ли такое множество B, согласованное с фрагментом b, что после присоединения его характеристической функции к программе p эта программа даёт на входе x какой-то из ответов «да» и «нет». Если такого множества нет, то вообще заботиться не о чем — утверждение леммы будет верным, если просто положить $a' = a$, $b' = b$.

Пусть такое множество B существует. Проследим за работой программы p на входе x для этого множества B. Прежде чем выдать свой ответ, программа может некоторое конечное число раз вызывать характеристическую функцию множества B. Возьмём фрагмент b', с которым B согласовано, и притом достаточно длинный, чтобы покрыть и зафиксировать все те места, к которым обращалась программа p. Тогда программа p будет давать тот же самый ответ не только для множества B, но и для любых множеств, согласованных с b'. Остаётся обеспечить, чтобы этот ответ был неверным, что можно сделать, включив x в область определения a' и выбрав $a'(x)$ противоречащим этому ответу. Лемма доказана.

Осталось лишь доказать утверждение теоремы, относящееся к $\mathbf{0}'$-вычислимости, для чего надо убедиться, что построение a' и b' в доказательстве леммы можно сделать $\mathbf{0}'$-алгоритмическим. Ключевой момент здесь — ответ на сформулированный при доказательстве леммы вопрос. Конечно, буквально перебрать континуум возможных множеств B, согласованных с фрагментом b, невозможно. Но это и не требуется — надо просто просматривать все варианты работы программы p. Когда она задаёт вопрос про не входящее в b число, просмотр разветвляется на два направления в зависимости от двух возможностей. Получается ветвящееся дерево вариантов, и вопрос состоит в том, получается ли ответ «да» или «нет» хоть на какой-то ветви. А этот вопрос можно переформулировать как вопрос о том, остановится ли некоторая программа (а именно, программа, просматривающая параллельно все ветви и останавливающаяся, как только на одной из них появится ответ «да» или «нет»).

Это замечание и завершает доказательство теоремы. $\triangleright$

Гораздо более сложен вопрос о том, существуют ли не просто $\mathbf{0}'$-разрешимые несравнимые по Тьюрингу множества, а перечислимые несравнимые по Тьюрингу множества. Эту проблему (так называемую *проблему Поста*) независимо решили американский математик Фридберг и Альберт Абрамович Мучни́к; интересно, что для построения перечислимых несравнимых множеств они использовали один и тот же подход, который получил название «метод приоритета».

7.6. Теорема Мучника–Фридберга: общая схема конструкции

Теорема 50. Существуют несравнимые по Тьюрингу перечислимые множества.

Мы приводим доказательство этой теоремы как пример более изощрённой техники, используемой в теории вычислимых функций. Однако надо иметь в виду, что в 1960-ые и 1970-ые годы передний край этой области ушёл далеко за горизонт, и приводимое ниже рассуждение стало скорее образцом простоты, чем сложности.

◁ Итак, мы хотим построить два перечислимых множества, ни одно из которых не сводится к другому по Тьюрингу. Мы будем строить их по шагам; на каждом шаге будет известна лишь конечная часть будущих множеств. Нам будет удобна такая терминология.

Будем называть *элементом* произвольную пару конечных множеств $\langle A, B\rangle$ натуральных чисел. Будем говорить, что элемент $\langle A', B'\rangle$ *продолжает* элемент $\langle A, B\rangle$, если $A \subset A'$ и $B \subset B'$. Мы построим вычислимую последовательность элементов, каждый из которых продолжает предыдущий; в пределе (объединении) они дадут искомые перечислимые несравнимые множества.

Будем называть *указанием* четвёрку конечных множеств $\langle A^+, A^-, B^+, B^-\rangle$, в которой A^+ не пересекается с A^- и B^+ не пересекается с B^-. Слово «указание» объясняется тем, что такие четвёрки указывают, чего мы хотим от элементов: A^+ — это числа, которые должны входить в A, а A^- — числа, которые не должны входить в A; аналогично для B. Формально, мы говорим, что элемент $\langle A, B\rangle$ *согласован* с указанием $\langle A^+, A^-, B^+, B^-\rangle$, если $A^+ \subset A$, $A^- \cap A = \varnothing$, $B^+ \subset B$, $B^- \cap B = \varnothing$. Будем говорить, что указание u_2 *сильнее* указания u_1, если всякий элемент, согласованный с u_2, согласован и с u_1 (то есть каждая из четырёх частей указания может только увеличиться).

Пусть $\alpha(X, Y)$ — произвольное свойство пары множеств $X, Y \subset \mathbb{N}$. С каждым таким свойством свяжем некоторую игру двух персонажей — Руководителя (Р) и Исполнителя (И). Игра происходит так: вначале И предъявляет Р некоторое указание u_0 и некоторый элемент e_0, согласованный с u_0. Мы будем называть их начальным указанием и начальным элементом. (Как мы увидим, в окончательной конструкции руководителей будет несколько, и начальное указание и элемент достаются свеженазначенному руководителю от его предшественников — но об этом дальше). Р отвечает некоторым указанием u_1, после этого И выбирает согласованный с ним элемент e_1, затем Р выбирает u_2, И выбирает e_2 и так далее (игра продолжается бесконечно).

Правила игры таковы:

- Каждый следующий выбираемый И элемент должен продолжать предыдущий (и потому все они продолжают начальный); он также должен быть согласован с последним указанием Р, но может не быть согласован с его предыдущими указаниями.

- Все указания Р должны быть сильнее начального указания (но не обязаны быть сильнее его предыдущих указаний!).

- Если очередное указание Р вызывает пат (то есть у И нет элемента, который был с ним согласован и продолжал предыдущий элемент), то игра заканчивается и Р проигрывает.

- Если игра бесконечна, то мы считаем Р победителем при выполнении двух условий. Первое из них состоит в том, что указания Р, начиная с некоторого момента игры, не меняются.

- Наконец, второе условие состоит в том, что предельные множества X и Y удовлетворяют условию $\alpha(X, Y)$, о котором мы говорили до начала описания игры. (Если i-ый элемент e_i есть $\langle X_i, Y_i\rangle$, то X и Y есть объединения возрастающих цепочек множеств $X_0 \subset X_1 \subset \ldots$ и $Y_0 \subset Y_1 \subset \ldots$)

Будем называть условие *выигрышным*, если существует вычислимая (реализуемая алгоритмом) стратегия для Р, гарантирующая его выигрыш. Дальнейший план действий такой. Мы покажем, что для любой программы p с вызовами внешней процедуры условие «p с оракулом для Y не разрешает X» является выигрышным. (Это рассуждение в значительной мере повторяет рассуждение из теоремы Клини – Поста, но несколько более сложно.) Более того, мы установим, что соответствующая стратегия вычислимо зависит от p.

С другой стороны, мы покажем, что для любого числа выигрышных условий α_i, для которых стратегии можно выбрать вычислимо зависящими от i, можно найти пару перечислимых множеств, удовлетворяющую всем условиям. Именно это последнее рассуждение будет использовать идею «приоритета»: у нас будет один исполнитель и счётное число руководителей, которым присвоены разные уровни приоритета (главный, менее главный, ещё менее главный и т. д.).

7.7. Теорема Мучника – Фридберга: выигрышные условия

Итак, пусть фиксирована программа p, которой Р хочет помешать разрешать множество X относительно множества Y. Что он должен для этого делать? (Мы будем описывать всё происходящее с его точки зрения.)

В начале Р получает некоторое указание и некоторый элемент, с ним согласованный. Все дальнейшие указания должны быть сильнее этого — мы всегда должны указывать включить определённые числа в X и Y и не включать некоторые другие (и тех, и других — конечное множество). Кроме того, есть некоторый начальный элемент — начальная пара множеств. Со временем И увеличивает эти множества по собственному усмотрению;

единственное, как мы можем на это повлиять — давая указания.

Итак, что же мы делаем? На первом шаге выберем какое-то число x, не входящее в начальное значение X и не затронутое начальным указанием. В нашем первом указании мы попросим не включать это x в X, то есть добавим x во вторую компоненту начального указания, которую мы когда-то обозначали A^-. (Если бы мы хотели, чтобы программа не разрешала Y, мы бы действовали симметрично и добавляли бы число в четвёртую компоненту, которая обозначалась B^-.)

Что это нам даёт? Если мы будем и дальше дублировать первое указание, то мы добьёмся, чтобы число x не принадлежало предельному множеству X. Но если в какой-то момент мы передумаем и захотим, чтобы x принадлежало X, это можно — достаточно изъять x из A^- и добавить его в A^+, что не вызовет пата. (Заметим, что это новое указание не будет сильнее прежнего — но по-прежнему будет сильнее начального, а только это и требуется.)

Так или иначе, мы выбрали такое x, сформировали первое указание и дублируем его, пока не видим причин изменить своё мнение. Причины эти могут состоять в следующем. На n-ом ходу игры мы выполняем n шагов работы программы p на входе x. (Напомним, что p — та самая программа, которой мы хотим не дать разрешать X с оракулом для Y.) При этом, когда программа вызывает внешнюю процедуру для Y, мы даём ответы в соответствии с текущим состоянием Y (то есть в соответствии с последним элементом, написанным И). Представим себе, что действительно за n шагов получился какой-то результат. Тогда мы пробуждаемся и смотрим, принадлежность и непринадлежность каких элементов множеству Y была при этом использована, и фиксируем текущее положение дел в нашем следующем и всех последующих (больше они меняться не будут) указаниях. Тем самым будет гарантирован тот же ответ программы p на предельном множестве Y. С другой стороны, мы можем добиться, чтобы x принадлежало X или нет по желанию (чтобы не принадлежало, не надо делать ничего, чтобы принадлежало — перенесём его в положительную часть указания, см. выше). Сделаем это так, чтобы ответ программы p стал неправильным.

Покажем, что эта стратегия действительно выигрышная. Есть два случая. Если мы в какой-то момент пробудились, то по построению программа p даёт неправильный ответ на числе x. Если же мы так и не пробудились, то программа p не даёт на x никакого ответа (при предельном значении оракула). Почему? В самом деле, любой такой ответ зависит от конечного числа вопросов к оракулу и требует конечного числа шагов работы — так что на достаточно далёком шаге игры, когда все нужные числа в оракуле появятся и времени на вычисление будет достаточно, мы должны были бы пробудиться.

Вычислимость нашей стратегии (в том числе вычислимая зависимость от параметра, то есть от программы p) очевидна. Осталось объяснить лишь, почему число различных указаний будет конечно. На самом деле их может быть максимум два — если мы когда-то пробуждались, чтобы далее заснуть навеки (а если не пробуждались, то вообще одно).

7.8. Теорема Мучника–Фридберга: метод приоритета

Теперь можно забыть о конкретной природе элементов и указаний и показать, что если есть последовательность выигрышных условий $\alpha_1, \alpha_2, \ldots$, причём выигрышные стратегии для Р вычислимо зависят от i, то есть пара перечислимых множеств, удовлетворяющая всем условиям.

Для этого представим себя на месте исполнителя, над которым есть последовательность руководителей, приоритеты которых убывают (первый — самый важный, второй — менее важный и т. д.). Каждый из руководителей отвечает за своё условие и имеет свою выигрышную стратегию. Вначале мы начинаем играть с первым руководителем и выполнять его указания. Когда они начинают повторяться, мы надеемся, что первый руководитель уже больше менять их не будет, и передаём текущий элемент и текущее указание в качестве начальных второму руководителю, и играем с ним. (Все указания второго будут сильнее временно стабилизировавшихся указаний первого.) Когда указания второго тоже стабилизируются, можно подключить третьего и т. д.

Что же происходит, если какой-то из руководителей неожиданно изменил своё указание? В этом случае мы следуем его указанию, не обращая внимания на указания менее приоритетных, а перед ними извиняемся и говорим, что пригласили их слишком рано. Но затем мы снова постепенно приглашаем их по той же схеме.

Покажем, что каждый руководитель рано или поздно получит возможность бессменно руководить. В самом деле, первый вообще ничего не знает о других, поскольку его указания самые важные и всегда исполняются. Значит, в некоторый момент он перестанет давать новые указания. Запущенная после этого инкарнация второго руководителя уже не встретит никаких помех, поскольку лишь изменение указаний первого может ей помешать, поэтому и указания второго стабилизируются, и т. д. При этом каждый руководитель добьётся выполнения своего свойства.

Теперь видно, почему в определении игры были важны начальные элементы и условия — настоящий Руководитель должен добиваться своего, исходя из любой начальной ситуации (кто знает, до чего доведут дело предыдущие!).

Вычислимость всех стратегий всех руководителей (плюс вычислимая зависимость от номера руководителя) гарантирует вычислимость описанного процесса, так что получатся перечислимые множества, которые удовлетворяют всем условиям.

Это рассуждение завершает доказательство теоремы Мучника – Фридберга. $\triangleright$

61. Покажите, что существует счётное число перечислимых множеств, никакие два их которых не сравнимы по Тьюрингу.

8. Арифметическая иерархия

8.1. Классы Σ_n и Π_n

Мы уже говорили, что перечислимые множества можно эквивалентно определить как проекции разрешимых множеств: множество $A \subset \mathbb{N}$ перечислимо тогда и только тогда, когда существует разрешимое множество $B \subset \subset \mathbb{N} \times \mathbb{N}$, проекцией которого оно является. Если отождествлять множества со свойствами, то можно сказать, что свойство $A(x)$ натуральных чисел перечислимо тогда и только тогда, когда его можно представить в виде

$$A(x) \Leftrightarrow \exists y B(x, y),$$

где $B(x, y)$ — некоторое разрешимое свойство.

(В этом разделе мы предполагаем знакомство читателя с простейшими логическими обозначениями: квантор $\exists x$ читается как «существует x», квантор $\forall x$ читается как «для всех x», знак $\wedge$ читается как «и» и называется конъюнкцией, знак $\vee$ читается как «или» и называется дизъюнкцией, знак $\neg$ читается как «неверно, что» и называется отрицанием. Как и раньше, знак $\Leftrightarrow$ означает равносильность.)

Возникает естественный вопрос: что можно сказать про другие наборы кванторов? Например, какие свойства представимы в виде

$$A(x) \Leftrightarrow \exists y \exists z C(x, y, z),$$

где C — разрешимое свойство троек натуральных чисел? Легко сообразить, что это по-прежнему перечислимые множества. В самом деле, два подряд идущих квантора одного вида можно заменить одним, использовав вычислимую нумерацию пар (которую мы обозначаем квадратными скобками): свойство C', для которого $C'(x, [y, z]) \Leftrightarrow C(x, y, z)$, также разрешимо, и $A(x) \Leftrightarrow \Leftrightarrow \exists w C'(x, w)$.

Другой вопрос: какие свойства представимы в виде

$$A(x) \Leftrightarrow \forall y B(x, y),$$

где $B(x, y)$ — некоторое разрешимое свойство? Ответ: те, отрицания которых перечислимы (как иногда говорят, *коперечислимые*). В самом деле, переходя к отрицаниям, имеем

$$\neg A(x) \Leftrightarrow \neg\forall y B(x, y) \Leftrightarrow \exists y(\neg B(x, y)),$$

а разрешимые свойства остаются разрешимыми при переходе к отрицаниям.

Дадим общее определение. Свойство A *принадлежит классу* Σ_n, если его можно представить в виде

$$A(x) \Leftrightarrow \exists y_1 \forall y_2 \exists y_3 \ldots B(x, y_1, y_2, \ldots, y_n)$$

(в правой части стоит n чередующихся кванторов) для некоторого разрешимого свойства B. Если в правой части поставить n чередующихся кванторов, начиная с квантора всеобщности $\forall$, то получится определение *класса* Π_n.

Отметим два свойства, которые мы по существу уже доказали:

Теорема 51. (а) Определение класса Σ_n [Π_n] не изменится, если в правой части разрешить большее число кванторов и требовать, чтобы первый квантор был квантором существования [всеобщности] и число групп одинаковых стоящих рядом кванторов равнялось n. (б) Отрицания свойств из класса Σ_n принадлежат классу Π_n и наоборот.

$\triangleleft$ Для доказательства первого утверждения достаточно соединять рядом стоящие одинаковые кванторы с помощью нумерации пар. Для доказательства второго надо проносить отрицание внутрь (меняя тип квантора), пока оно не окажется у разрешимого свойства (где оно роли не играет). $\triangleright$

Мы говорили о свойствах; на языке множеств можно сказать так: множества класса Σ_n получаются из разрешимых с помощью последовательности операций «проекция – дополнение – проекция – дополнение – ... – проекция», в которой всего n операций проекции. Каждая операция проекции уменьшает размерность множества (число аргументов у свойства) на единицу, так что начинать надо с разрешимых подмножеств $\mathbb{N}^{n+1}$.

Теорема 52. Пересечение и объединение двух множеств из класса Σ_n принадлежит Σ_n. Пересечение и объединение двух множеств из класса Π_n принадлежит Π_n.

$\triangleleft$ Удобно выразить это утверждение на логическом языке, сказав, что конъюнкция и дизъюнкция любых двух свойств из класса Σ_n лежат в том же классе (аналогично для Π_n). На этом же языке удобно провести и доказательство: если, скажем,

$$A(x) \Leftrightarrow \exists y \forall z B(x,y,z),$$
$$C(x) \Leftrightarrow \exists u \forall v D(x,u,v),$$

то

$$A(x) \wedge C(x) \Leftrightarrow \exists y \exists u \forall z \forall v [B(x,y,z) \wedge D(x,u,v)],$$

записанное в квадратных скобках свойство разрешимо и остаётся только соединить пары кванторов в один, как объяснялось выше. Аналогично можно действовать для классов Σ_n и Π_n при произвольном n. $\triangleright$

Мы определяли классы Σ_n и Π_n для множеств натуральных чисел; аналогичным образом это можно сделать и для множеств пар натуральных чисел, троек и вообще любых «конструктивных объектов». Заметим, что проекция множества пар, принадлежащего классу Σ_n, также принадлежит Σ_n (поскольку два квантора существования в начале можно объединить в один).

Добавляя фиктивные кванторы, легко убедиться, что каждый из двух классов Σ_n и Π_n содержится в каждом из классов Σ_{n+1} и Π_{n+1}. Можно написать ещё так:

$$\Sigma_n \cup \Pi_n \subset \Sigma_{n+1} \cap \Pi_{n+1}.$$

Теорема 53. Классы Σ_n и Π_n «наследственны вниз» относительно m-сводимости: если $A \leqslant_m B$ и $B \in \Sigma_n$ [$B \in \Pi_n$], то и $A \in \Sigma_n$ [$A \in \Pi_n$].

$\triangleleft$ В самом деле, пусть A сводится к B с помощью всюду определённой вычислимой функции f, то есть $x \in A \Leftrightarrow f(x) \in B$. Пусть B принадлежит, например, классу Σ_3, то есть

$$x \in B \Leftrightarrow \exists y \forall z \exists u R(x, y, z, u),$$

где R — некоторое разрешимое свойство. Тогда

$$x \in A \Leftrightarrow f(x) \in B \Leftrightarrow \exists y \forall z \exists u R(f(x), y, z, u),$$

и осталось заметить, что $R(f(x), y, z, u)$ (как свойство четвёрки $\langle x, y, z, u\rangle$) разрешимо. $\rhd$

62. Докажите, что если множество A принадлежит классу Σ_n, то множество $A \times A$ также принадлежит этому классу.

63. Докажите, что если множества A и B принадлежат классу Σ_n, то их разность $A \setminus B$ принадлежит классу $\Sigma_{n+1} \cap \cap \Pi_{n+1}$.

8.2. Универсальные множества в Σ_n и Π_n

До сих пор мы не показали, что классы Σ_n и Π_n действительно различаются при разных n. Чтобы показать это, убедимся, что в каждом из этих классов имеется универсальное множество (для соответствующего класса) и что оно не принадлежит меньшим классам.

Теорема 54. Для любого n в классе Σ_n существует множество, универсальное для всех множеств класса Σ_n. (Его дополнение будет универсальным в классе Π_n.)

Говоря об универсальном множестве из класса Σ_n, мы имеем в виду множество пар натуральных чисел, которое принадлежит классу Σ_n и среди сечений которого встречаются все множества натуральных чисел, принадлежащие классу Σ_n.

$\lhd$ Для класса Σ_1 (перечислимых множеств) существование универсального множества мы уже обсуждали. С его помощью можно построить универсальные множества и для более высоких классов иерархии. (Начинать надо с первого уровня, так как на «нулевом» уровне не существует универсального разрешимого множества.)

По определению свойства класса Π_2 имеют вид $S(x) \Leftrightarrow \Leftrightarrow \forall y \exists z R(x, y, z)$, где R — некоторое разрешимое свойство. Но их можно эквивалентно определить и как свойства вида $S(x) \Leftrightarrow \forall y P(x, y)$, где P — некоторое перечислимое свойство. Теперь уже видно, как построить универсальное множество класса Π_2. Возьмём универсальное перечислимое свойство $U(n,x,y)$, из которого фиксацией различных n получаются все перечислимые свойства пар натуральных чисел. Тогда из свойства $T(n,x) = \forall y U(n,x,y)$ при различных натуральных n получаются все Π_2-свойства натуральных чисел. С другой стороны, само свойство T по построению принадлежит классу Π_2.

Дополнение к универсальному Π_2-множеству будет, очевидно, универсальным Σ_2-множеством.

Аналогично можно действовать и для Σ_3- и Π_3-множеств (удобнее сначала рассуждать о Σ_3-множествах, так как в них внутренний квантор является квантором существования и задаёт перечислимое множество), и вообще для Σ_n- и Π_n-множеств. $\triangleright$

Теорема 55. Универсальное Σ_n-множество не принадлежит классу Π_n. Аналогичным образом, универсальное Π_n-множество не принадлежит классу Σ_n.

$\triangleleft$ Рассмотрим универсальное Σ_n-свойство $T(m,x)$. По определению это означает, что среди его сечений (получающихся, если зафиксировать m) есть все Σ_n-свойства. Пусть T принадлежит классу Π_n. Тогда его диагональ, свойство $D(x) = T(x,x)$, также принадлежит классу Π_n (например, потому, что $D \leqslant_m T$), а её отрицание, свойство $\neg D(x)$, принадлежит классу Σ_n. Но этого не может быть, так как $\neg D$ отлично от всех сечений свойства T (оно отличается от m-го сечения в точке m), а T универсально. $\triangleright$

Из этой теоремы следует, в частности, что любой из классов Σ_n и Π_n является собственным подмножеством любого из классов Σ_{n+1} и Π_{n+1}. (Мы увидим вскоре, что даже объединение $\Sigma_n \cup \Pi_n$ является собственным подмножеством пересечения $\Sigma_{n+1} \cap \Pi_{n+1}$.)

8.3. Операция скачка

Мы хотим показать, что класс Σ_n совпадает с классом всех A-перечислимых множеств для некоторого множества A (зависящего от n, естественно). Чтобы объяснить, что это за множество, нам понадобится так называемая операция скачка.

Пусть X — произвольное множество. Среди X-перечислимых множеств есть универсальное. Это множество будет m-полным в классе X-перечислимых множеств в том смысле, что все другие X-перечислимые множества к нему m-сводятся. Сводящая функция, как мы видели, имеет вид $x \mapsto [n, x]$ (и вычислима безо всякого оракула, как того и требует определение m-сводимости). Будем обозначать через X' любое m-полное множество в классе X-перечислимых множеств. Можно сказать, что X' определено с точностью до m-эквивалентности.

Более формально, будем говорить, что множества P и Q являются *m-эквивалентными*, если $P \leqslant_m Q$ и $Q \leqslant_m P$. (Легко видеть, что это действительно отношение эквивалентности.) Класс эквивалентных множеств называют *m-степенью*. Таким образом, можно сказать, что мы для каждого множества X определили некоторую m-степень X'.

Аналогичным образом определяют *T-степени* (которые называют также *тьюринговыми степенями* или *степенями неразрешимости*) как классы T-эквивалентных множеств; множества P и Q называют *T-эквивалентными*, или *эквивалентными по Тьюрингу*, если $P \leqslant_T Q$ и $Q \leqslant_T P$, то есть если каждое из множеств разрешимо относительно другого. Если множества P и Q эквивалентны по Тьюрингу, то класс P-вычислимых функций совпадает с классом Q-вычислимых функций (а класс P-перечислимых множеств совпадает с классом Q-перечислимых множеств). Введя понятие T-степени, можно сказать, что m-степень X' определяется T-степенью множества X и тем самым определено отображение множества всех T-степеней в множество всех m-степеней. Это отображение называют *операцией скачка*; множество (точнее, m-степень) X' называют *скачком* множества (точнее, T-степени) X.

64. Могут ли при этом отображении разные T-степени переходить в одну и ту же m-степень?

65. Докажите, что любые два m-полных в классе Σ_n множества вычислимо изоморфны (отличаются вычислимой перестановкой).

66. Покажите, что для любого перечислимого множества A можно указать такое действительное число α, что множество всех рациональных чисел, меньших α, будет перечислимо и эквивалентно по Тьюрингу множеству A.

Обычно, впрочем, операцию скачка рассматривают на T-степенях, считая её результатом T-степень, содержащую X' (это законно, так как T-классификация более грубая).

Нам понадобятся следующие T-степени: $\mathbf{0}$ (степень, содержащая все разрешимые множества), $\mathbf{0}'$ (её скачок, степень m-полного перечислимого неразрешимого множества; мы её уже рассматривали), затем $\mathbf{0}''$ (скачок степени $\mathbf{0}'$), $\mathbf{0}'''$ и так далее; вообще $\mathbf{0}^{(n+1)} = (\mathbf{0}^{(n)})'$.

Теорема 56. При любом $n \geqslant 1$ класс Σ_n совпадает с классом всех $\mathbf{0}^{(n-1)}$-перечислимых множеств.

(Пока что мы знаем это при $n = 1$.)

$\triangleleft$ Докажем сначала, что все Σ_n-множества перечислимы относительно $\mathbf{0}^{(n-1)}$. Это делается индукцией по n. При $n = 1$ это известно. Рассмотрим теперь произвольное множество X из Σ_2. По определению,

$$x \in X \Leftrightarrow \exists y \forall z R(x, y, z),$$

где R — разрешимое свойство. Свойство $\forall z R(x, y, z)$ имеет перечислимое отрицание. Это отрицание разрешимо относительно $\mathbf{0}'$, так как m-сводится к m-полному перечислимому множеству. Значит, и само свойство $\forall z R(x, y, z)$ разрешимо относительно $\mathbf{0}'$. Поэтому его проекция, множество X, перечислимо относительно $\mathbf{0}'$.

Аналогично можно рассуждать и для бо́льших n. Если X принадлежит Σ_3, то

$$x \in X \Leftrightarrow \exists y R(x, y),$$

где R принадлежит Π_2. Отрицание R принадлежит Σ_2 (по доказанному), поэтому $\mathbf{0}'$-перечислимо, поэтому $\mathbf{0}''$-разрешимо, поэтому само R тоже $\mathbf{0}''$-разрешимо, а его проекция $\mathbf{0}''$-перечислима.

Первая половина теоремы доказана.

Для доказательства второй половины нам потребуется некоторое свойство классов Σ_n и Π_n. Рассмотрим какую-нибудь вычислимую нумерацию всех конечных множеств натуральных чисел. Обозначим через D_x конечное множество номер x. Для произвольного множества A рассмотрим множество $\mathrm{Subset}(A)$ всех конечных подмножеств A, точнее, множество всех их номеров:

$$x \in \mathrm{Subset}(A) \Leftrightarrow D_x \subset A.$$

Лемма 1. Если множество A принадлежит классу Σ_n [или Π_n], то множество $\mathrm{Subset}(A)$ также принадлежит классу Σ_n [соответственно Π_n].

(Утверждение этой леммы обобщает сформулированное в задаче 62 утверждение о множестве $A \times A$: теперь мы рассматриваем не пары, а произвольные кортежи.)

Доказательство леммы. Пусть множество A принадлежит, например, классу Σ_3:

$$x \in A \Leftrightarrow \exists y \forall z \exists t\, R(x, y, z, t),$$

где R — разрешимое свойство. Тогда можно записать свойство $\{x_1, \ldots, x_n\} \subset A$ следующим образом:

$$\exists \langle y_1, \ldots, y_n \rangle \forall \langle z_1, \ldots, z_n \rangle \exists \langle t_1, \ldots, t_n \rangle [R(x_1, y_1, z_1, t_1) \wedge \ldots \\ \ldots \wedge R(x_n, y_n, z_n, t_n)]$$

Эта формула использует кванторы по кортежам натуральных чисел (переменной длины), но их можно заменить на номера этих кортежей в какой-нибудь вычислимой нумерации. При этом стоящая под кванторами формула (она записана несколько условно: символическая конъюнкция на самом деле имеет переменную длину) является разрешимым свойством номеров кортежей, поэтому вся правая часть является Σ_n-свойством.

(На самом деле мы допустили ещё одну вольность речи: правая часть является не свойством конечного множества $\{x_1, \ldots, x_n\}$, а свойством кортежа (упорядоченной последовательности) $\langle x_1, \ldots, x_n \rangle$. Но переход от номера множества к номеру какого-то кортежа, содержащего все его элементы, вычислим, так что проблемы тут нет.)

Лемма доказана.

67. Докажите, что если A принадлежит классу Σ_n $[\Pi_n]$, то и множество $\mathrm{Intersect}(A)$ номеров конечных множеств, пересекающихся с A, принадлежит классу Σ_n $[\Pi_n]$.

68. Пусть свойство $R(x,y)$ пар натуральных чисел принадлежит классу Σ_n. Покажите, что свойство

$$S(x) = (\forall y \leqslant x)\, R(x,y)$$

принадлежит Σ_n. (Ограниченный квантор $(\forall y \leqslant x)$ читается как «для всех y, не превосходящих x».)

Переходя к дополнениям, мы немедленно получаем такое утверждение:

Лемма 2. Если A принадлежит классу Σ_n $[\Pi_n]$, то множество $\mathrm{Disjoint}(A)$, состоящее из номеров конечных множеств, не пересекающихся с A, принадлежит Π_n $[\Sigma_n]$.

Доказательство леммы. Не пересекаться с A означает быть подмножеством дополнения к A, и остаётся воспользоваться предыдущей леммой и тем, что дополнение к множеству из класса Σ_n $[\Pi_n]$ лежит в классе Π_n $[\Sigma_n]$. Лемма доказана.

Теперь мы можем перейти к доказательству того факта, что все множества, перечислимые относительно $\mathbf{0}^{(n-1)}$, принадлежат классу Σ_n. Это также доказывается индукцией по n.

Начнём с первого нетривиального случая: почему множество, перечислимое относительно $\mathbf{0}'$, лежит в Σ_2? (Здесь можно было бы применить критерий $\mathbf{0}'$-вычислимости, приведённый выше, но мы предпочитаем действовать по общей схеме, которая годится и для бóльших n.)

Итак, пусть некоторое множество A перечислимо относительно $\mathbf{0}'$. Тогда оно перечислимо относительно некоторого перечислимого множества B, то есть перечислимо относительно характеристической функции b множества B. Согласно доказанному нами выше критерию (теорема 45, с. 79), это означает, что существует перечислимое множество Q пар вида $\langle x,t\rangle$, где x — число, а

t — образец, для которого

$$x \in A \Leftrightarrow \exists t[(\langle x,t\rangle \in Q) \text{ и } (b \text{ продолжает } t)].$$

Образец t представляет собой функцию, определённую на конечном множестве и принимающую значения 0 и 1. (Если у t есть какие-то другие значения, то он не может быть частью характеристической функции множества B и роли не играет.) Условие «b продолжает t» в терминах множества B звучит так: B содержит множество тех аргументов, на которых t принимает значение 1, и не пересекается с множеством тех аргументов, на которых t принимает значение 0. Поэтому вместо образцов можно говорить о парах конечных множеств; тогда вместо Q надо рассмотреть перечислимое множество P троек вида $\langle x,u,v\rangle$ и написать так:

$$x \in A \Leftrightarrow \exists u \exists v[(\langle x,u,v\rangle \in P) \text{ и } (D_u \text{ содержится в } B) \text{ и } (D_v \text{ не пересекается с } B)].$$

Теперь вместо «D_u содержится в B» напишем «$u \in \in \mathrm{Subset}(B)$», а вместо «$D_v$ не пересекается с B» напишем «$v \in \mathrm{Disjoint}(B)$». Остаётся заметить, что все три свойства, соединённые союзом «и» в правой части, принадлежат классу Σ_2 и даже меньшим классам. Именно, первые два принадлежат классу Σ_1, так как P и B перечислимы (для второго свойства применяем лемму 1). Третье же принадлежит классу Π_1 по лемме 2. Поэтому их конъюнкция принадлежит классу Σ_2, и операция проекции (кванторы $\exists u \exists v$) не выводит за пределы этого класса. Случай $n = 2$ разобран.

Далее, если какое-то множество A перечислимо относительно $\mathbf{0}''$, то по определению это означает, что оно перечислимо относительно некоторого B, которое перечислимо относительно $\mathbf{0}'$ и потому лежит в Σ_2. После этого все рассуждения проходят точно так же со сдвигом на 1. Аналогично разбираются и все следующие значения n. $\triangleright$

Из доказанной теоремы немедленно вытекает такое следствие:

Теорема 57. Пересечение классов $\Sigma_n \cap \Pi_n$ совпадает с классом разрешимых относительно $\mathbf{0}^{(n-1)}$ множеств.

$\triangleleft$ В самом деле, релятивизованная теорема Поста (теорема 2, с. 13) утверждает, что некоторое множество является X-разрешимым тогда и только тогда, когда оно и его дополнение X-перечислимы (здесь X — произвольный оракул). $\triangleright$

Теорема 58. Класс $\Sigma_n \cup \Pi_n$ является собственным подмножеством класса $\Sigma_{n+1} \cap \Pi_{n+1}$.

$\triangleleft$ Вспомним, что такое $\mathbf{0}^{(n)}$. Это — степень множества X, являющегося m-полным в классе $\mathbf{0}^{(n-1)}$-перечислимых множеств. Поскольку X является m-полным в указанном классе, оно не $\mathbf{0}^{(n-1)}$-разрешимо, то есть его дополнение не является $\mathbf{0}^{(n-1)}$-перечислимым.

Значит, по доказанной только что теореме X принадлежит классу Σ_n, а его дополнение — нет. Напротив, дополнение к X принадлежит Π_n, но не Σ_n. Рассмотрим теперь «соединение» множества X с его дополнением, т. е. множество

$$\{2n \mid n \in X\} \cup \{2n+1 \mid n \notin X\}.$$

К этому множеству m-сводятся как X, так и его дополнение, поэтому оно не может принадлежать ни Σ_n, ни Π_n. С другой стороны, оно, очевидно, разрешимо относительно X, поэтому по доказанной теореме принадлежит и Σ_{n+1}, и Π_{n+1}. $\triangleright$

8.4. Классификация множеств в иерархии

Интересно посмотреть, какое место разные конкретные множества занимают в описанной нами иерархии. Например, что можно сказать о множестве номеров какой-то фиксированной вычислимой функции в главной нумерации?

Мы уже говорили, что множество всех номеров всех функций с непустой областью определения перечислимо, то есть принадлежит классу Σ_1. Следовательно, его дополнение, множество всех номеров нигде не определённой функции, принадлежит классу Π_1. (Классу Σ_1 оно принадлежать не может, так как неразрешимо, см. теорему 21, с. 33.)

69. Докажите, что множество номеров нигде не определённой функции в любой главной нумерации является m-полным в классе Π_1 множеством.

А что можно сказать о номерах других функций? Например, что можно сказать о множестве номеров тождественно нулевой функции? Оказывается, можно получить в некотором смысле полный ответ на этот вопрос.

Теорема 59. (а) Пусть U — вычислимая универсальная функция для класса вычислимых функций. Тогда множество тех n, при которых U_n всюду определено и тождественно равно 0, принадлежит классу Π_2. (б) Пусть U — главная универсальная функция. Тогда указанное множество является m-полным в классе Π_2.

Заметим, что требование главности в пункте (б) существенно: в однозначной нумерации (см. с. 35) это множество состоит из единственного номера.

$\triangleleft$ Интересующее нас свойство числа n можно записать так: для любого k найдётся такое t, что *за t шагов вычисление значения $U(n,k)$ закончится и даст результат* 0. Выделенная часть является разрешимым свойством, а перед ней стоят два квантора как раз нужного вида. Итак, пункт (а) доказан.

Докажем утверждение (б). Пусть имеется произвольное множество P из класса Π_2. При этом

$$x \in P \Leftrightarrow \forall y \exists z R(x,y,z),$$

где R — некоторое разрешимое свойство. Рассмотрим теперь функцию $S(x,y)$, вычисляемую таким алгоритмом: перебирая все числа, ищем число z, для которого $R(x,y,z)$; как только (и если) такое число найдено, выдаём на выход 0. Ясно, что x-ое сечение S_x функции S будет тождественно нулевой функцией в том и только том случае, когда $x \in P$. Применим свойство главности и получим функцию s, для которой $U_{s(x)} = S_x$. Она и будет сводить P к множеству всех номеров тождественно нулевой функции. $\triangleright$

Что можно сказать про другие функции? Для любой вычислимой универсальной функции U и любой вычислимой функции f множество всех U-номеров функции f является Π_2-множеством. Верен даже более сильный факт: свойство $U_m = U_n$ (числа m и n являются номерами одной и той же функции) является Π_2-свойством пары $\langle m, n\rangle$, поэтому тем более любое его сечение (т. е. множество всех номеров любой конкретной функции) является Π_2-множеством. В самом деле, свойство $U_m = U_n$ можно сформулировать так: «для всяких x и t_1 найдётся такое t_2, что *если вычисление $U(m,x)$ завершается за t_1 шагов, то вычисление $U(n,x)$ завершается за t_2 шагов с тем же результатом, и наоборот*». Выделенная часть разрешима, а до неё стоит Π_2-префикс.

Можно даже понять, для каких функций множество номеров является Π_2-полным: для функций с бесконечной областью определения. Если область определения функции конечна, то множество всех её номеров является $\mathbf{0}'$-разрешимым (и потому не Π_2-полным): имея оракул для проблемы остановки, можно убедиться, что функция определена всюду, где она должна быть определена (после чего проверить, что значения правильны), а затем проверить, что ни в одной из оставшихся точек она не определена (процесс поиска точки вне данного конечного множества, в которой функция определена, является перечислимым процессом и потому его успешность может быть проверена с помощью $\mathbf{0}'$-оракула).

Если же функция имеет бесконечную область определения, то существует бесконечное разрешимое множество, во всех точках которого функция определена (задача 12, с. 15). Затем можно использовать по существу ту же конструкцию, что и для нулевой функции, но только внутри этого подмножества (не трогая элементов вне него).

70. Проведите это рассуждение подробно.

71. Покажите, что множество всех номеров всех всюду определённых функций (в главной нумерации) является Π_2-полным.

72. В каком наименьшем классе арифметической иерархии лежит множество номеров всех функций с бесконечной областью определения? Будет ли оно m-полным в этом классе?

73. Покажите, что для любой (не обязательно главной!) нумерации множество всех номеров всюду определённых функций неперечислимо. Более того, оно не имеет перечислимого подмножества, включающего в себя хотя бы одному номеру каждой вычислимой всюду определённой функции. (Указание: используйте диагональную конструкцию.)

В книге Х. Роджерса ([8], параграф 14.8) приведено много результатов подобного рода для многих других свойств вычислимых функций и перечислимых множеств. Например, для любого m-полного множества K множество всех его номеров является Π_2-полным. (Здесь и далее, говоря о номерах, мы имеем в виду главную нумерацию перечислимых множеств.) Множество номеров всех конечных множеств является Σ_2-полным. Множество номеров множеств, содержащих хотя бы один номер бесконечного множества, является Σ_3-полным. Множество номеров всех разрешимых множеств является Σ_3-полным. Множество номеров всех множеств с конечными дополнениями является Σ_3-полным.

74. Докажите перечисленные утверждения (или прочтите их доказательства в книге Роджерса [8]).

75. Рассмотрим квантор $\exists^\infty x$, который читается как «существует бесконечно много x, для которых». Покажите, что все свойства из класса Π_2 (и только они) представимы в виде

$$\exists^\infty x\,(\text{разрешимое свойство})$$

и что все свойства из класса Σ_3 (и только они) представимы в виде

$$\exists^\infty x\,\forall y\,(\text{разрешимое свойство}).$$

(Аналогичное утверждение верно и для старших классов, см. теорему XVIII в разделе 14.8 книги [8].)

9. Машины Тьюринга

9.1. Зачем нужны простые вычислительные модели?

До сих пор нам было удобно ссылаться на программистский опыт, говоря об алгоритмах, программах, интерпретаторах, пошаговом выполнении и т. д. Это позволяло нам игнорировать детали построения тех или иных алгоритмов под тем предлогом, что читатель их легко восстановит (или хотя бы поверит — всё-таки не каждый читатель в своей жизни писал интерпретатор паскаля на паскале).

Но в некоторых случаях этого недостаточно. Пусть, например, мы хотим доказать алгоритмическую неразрешимость какой-то задачи, в определении которой ничего не говорится о программах (в этом разделе, например, мы докажем неразрешимость проблемы равенства слов в полугруппах, заданных образующими и соотношениями). Это обычно делается так. Мы показываем, что проблема остановки сводится к этой задаче. Для этого мы моделируем работу произвольного алгоритма в терминах рассматриваемой задачи (что это значит, будет видно из приводимого ниже примера). При этом нам важно, чтобы определение алгоритма было как можно проще.

Таким образом, наш план таков. Мы опишем довольно просто определяемый класс машин (его можно выбирать по-разному, мы будем использовать так называемые машины Тьюринга), затем объявим, что всякая вычислимая функция может быть вычислена на такой машине, а затем покажем, что вопрос об остановке машины Тьюринга можно свести к вопросу о равенстве слов в полугруппе.

Другая причина, по которой важны простые вычислительные модели (таких моделей много — разные виды машин Тьюринга, адресные машины и т. п.), связана с теорией сложности вычислений, когда нас начинает интересовать время выполнения программ. Но этот вопрос выходит за рамки классической теории алгоритмов.

9.2. Машины Тьюринга: определение

Машина Тьюринга имеет бесконечную в обе стороны *ленту*, разделённую на квадратики (*ячейки*). В каждой ячейке может быть записан некоторый символ из фиксированного (для данной машины) конечного множества, называемого *алфавитом* данной машины. Один из символов алфавита выделен и называется «пробелом» — предполагается, что изначально вся лента пуста, то есть заполнена пробелами.

Машина Тьюринга может менять содержимое ленты с помощью специальной читающей и пишущей *головки*, которая движется вдоль ленты. В каждый момент головка находится в одной из ячеек. Машина Тьюринга получает от головки информацию о том, какой символ та видит, и в зависимости от этого (и от своего внутреннего состояния) решает, что делать, то есть какой символ записать в текущей ячейке и куда сдвинуться после этого (налево, направо или остаться на месте). При этом также меняется внутреннее состояние машины (мы предполагаем, что машина — не считая ленты — имеет конечную память, то есть конечное число внутренних состояний). Ещё надо договориться, с чего мы начинаем и когда кончаем работу.

Таким образом, чтобы задать машину Тьюринга, надо указать следующие объекты:

- произвольное конечное множество A (*алфавит*); его элементы называются *символами*;
- некоторый выделенный символ $a_0 \in A$ (*пробел*, или *пустой символ*);
- конечное множество S, называемое множеством *состояний*;
- некоторое выделенное состояние $s_0 \in S$, называемое *начальным*;
- *таблицу переходов*, которая определяет поведение машины в зависимости от состояния и текущего символа (см. ниже);

- некоторое подмножество $F \subset S$, элементы которого называются *заключительными состояниями* (попав в такое состояние, машина останавливается).

Таблица переходов устроена следующим образом: для каждой пары $\langle$текущее состояние, текущий символ$\rangle$ указана тройка $\langle$новое состояние, новый символ, сдвиг$\rangle$. Здесь сдвиг — одно из чисел -1 (влево), 0 (на месте) и 1 (направо). Таким образом, таблица переходов есть функция типа $S \times A \to S \times A \times \{-1, 0, 1\}$, определённая на тех парах, в которых состояние не является заключительным.

Остаётся описать поведение машины Тьюринга. В каждый момент имеется некоторая *конфигурация*, складывающаяся из содержимого ленты (формально говоря, содержимое ленты есть произвольное отображение $\mathbb{Z} \to A$), текущей позиции головки (некоторое целое число) и текущего состояния машины (элемент S). Преобразование конфигурации в следующую происходит по естественным правилам: мы смотрим в таблице, что надо делать для данного состояния и для данного символа, то есть выясняем новое состояние машины, меняем символ на указанный и после этого сдвигаем головку влево, вправо или оставляем на месте. При этом, если новое состояние является одним из заключительных, работа машины заканчивается. Остаётся договориться, как мы подаём информацию на вход машины и что считается результатом её работы. Будем считать, что алфавит машины, помимо пробела, содержит символы 0 и 1 (а также, возможно, ещё какие-то символы). Входом и выходом машины будут конечные последовательности нулей и единиц (двоичные слова). Входное слово записывается на пустой ленте, головка машины ставится в его первую клетку, машина приводится в начальное состояние и запускается. Если машина останавливается, результатом считается двоичное слово, которое можно прочесть, начиная с позиции головки и двигаясь направо (пока не появится символ, отличный от 0 и 1).

Таким образом, любая машина Тьюринга задаёт некоторую частичную функцию на двоичных словах. Все такие функции естественно назвать *вычислимыми на машинах Тьюринга*.

9.3. Машины Тьюринга: обсуждение

Разумеется, наше определение содержит много конкретных деталей, которые можно было бы изменить. Например, лента может быть бесконечной только в одну сторону. Можно придать машине две ленты. Можно считать, что машина может либо написать новый символ, либо сдвинуться, но не то и другое вместе. Можно ограничить алфавит, считая, скажем, что в нём должно быть ровно 10 символов. Можно потребовать, чтобы в конце на ленте ничего не было, кроме результата работы (остальные клетки должны быть пусты). Все перечисленные и многие другие изменения не меняют класса вычислимых на машинах Тьюринга функций. Конечно, есть и небезобидные изменения. Например, если запретить машине двигаться налево, то это радикально поменяет дело — по существу лента станет бесполезной, так как к старым записям уже нельзя будет вернуться.

Как понять, какие изменения безобидны, а какие нет? Видимо, тут необходим некоторый опыт практического программирования на машинах Тьюринга, хотя бы небольшой. После этого уже можно представлять себе возможности машины, не выписывая программы полностью, а руководствуясь лишь приблизительным описанием. В качестве примера опишем машину, которая удваивает входное слово (изготавливает слово XX, если на входе было слово X).

Если машина видит пробел (входное слово пусто), она кончает работу. Если нет, она запоминает текущий символ и ставит пометку (в алфавите помимо символов 0 и 1 будут ещё их «помеченные варианты» $\overline{0}$ и $\overline{1}$). Затем она движется направо до пустой клетки, после чего пишет там копию запомненного символа. Затем она движется налево до пометки; уткнувшись в пометку, отходит назад и запоминает следующий символ и так далее, пока не скопирует всё слово.

Имея некоторый опыт, можно за всеми этими фразами видеть конкретные куски программы для машины Тьюринга. Например, слова «запоминает символ и движется направо» означают, что есть две группы состояний, одна для ситуации, когда запомнен нуль, другая — когда запомнена единица, и внутри каждой группы запрограммировано движение направо до первой пустой клетки.

Имея ещё чуть больше опыта, можно понять, что в этом описании есть ошибка — не предусмотрен механизм остановки, когда всё слово будет скопировано, поскольку копии символов ничем не отличаются от символов исходного слова. Ясно и то, как ошибку исправить — надо в качестве копий писать специальные символы $\tilde{0}$ и $\tilde{1}$, а на последнем этапе все пометки удалить.

76. Покажите, что функция «обращение», переворачивающая слово задом наперёд, вычислима на машине Тьюринга.

Другой пример неформального рассуждения: объясним, почему можно не использовать дополнительных символов, кроме 0, 1 и пустого символа. Пусть есть машина с большим алфавитом из N символов. Построим новую машину, которая будет моделировать работу старой, но каждой клетке старой будет соответствовать блок из k клеток новой. Размер блока (число k) будет фиксирован так, чтобы внутри блока можно было бы закодировать нулями и единицами все символы большого алфавита. Исходные символы 0, 1 и пустой будем кодировать как 0, за которым идут $(k-1)$ пустых символов, 1, за которым идут $(k-1)$ пустых символов, и группу из k пустых символов. Для начала надо раздвинуть буквы входного слова на расстояние k, что можно сделать без дополнительных символов (дойдя до крайней буквы, отодвигаем её, затем дойдя до следующей, отодвигаем её и крайнюю и так далее); надо только понимать, что можно идентифицировать конец слова как позицию, за которой следует более k пустых символов. Ясно, что в этом процессе мы должны хранить в памяти некоторый конечный объём информации, так что это возможно. После этого уже можно моделировать работу исходной машины по шагам, и для этого тоже достаточно конечной памяти (т. е. конечного числа состояний), так как нам важна только небольшая окрестность головки моделируемой машины. Наконец, надо сжать результат обратно.

Утверждение о том, что всякая вычислимая функция вычислима на машине Тьюринга, называют *тезисом Тьюринга.* Конечно, его смысл зависит от того, что понимать под словами «вычислимая функция». Если понимать их в расплывчато-интуитивном смысле («функция вычисляется алгоритмически, то есть по чётким, недвусмысленным, однозначным правилам» или что-то в таком роде), конечно, ни о каком доказательстве тезиса Тьюринга не может быть речи. Можно лишь говорить, что многовековая практика человечества от Евклида до Кнута не встретилась с примером алгоритма, который нельзя было бы записать как программу машины Тьюринга и т. п. Впрочем, ещё один (не слишком убедительный) аргумент приведён ниже.

Но если понимать слово «вычислимая» в тезисе Тьюринга как «вычислимая с помощью программы на паскале» и представить себе на минуту, что синтаксис и семантика паскаль-программ точно определены, то тезис Тьюринга станет уже чётким утверждением, которое может быть истинным или ложным, и которое можно доказывать. Конечно, такое доказательство по необходимости должно использовать формальное описание синтаксиса и семантики паскаля, и потому никем не проводилось, но для более простых вычислительных моделей это действительно можно формально доказать. Впрочем, такого рода доказательства сродни доказательству корректности длинной программы, и потому желающих их писать и тем более читать немного.

В заключение обсуждения приведём обещанный выше аргумент в пользу того, что любая вычислимая функция вычислима на машине Тьюринга. Пусть есть функция, которую человек умеет вычислять. При этом, он, естественно, должен использовать карандаш и бумагу, так как количество информации, которое он может хранить «в уме», ограничено. Будем считать, что он пишет на отдельных листах бумаги. Помимо текущего листа, есть стопка бумаг справа и стопка слева; в любую из них можно положить текущий лист, завершив с ним работу, а из другой стопки взять следующий. У человека есть карандаш и ластик. Поскольку очень мелкие буквы на листе неразличимы, число отчётливо различных состояний листа конечно, так что можно считать, что в каждый момент на листе записана одна буква из некоторого конечного (хотя и весьма большого) алфавита. Человек тоже имеет конечную память, так что его состояние есть элемент некоторого конечного множества. При этом можно составить некоторую таблицу, в которой записано, чем кончится его работа над листом с данным содержимым, начатая в данном состоянии (что будет на листе, в каком состоянии будет человек и из какой пачки будет взят следующий лист). Теперь уже видно, что действия человека как раз соответствуют работе машины Тьюринга с большим (но конечным) алфавитом и большим (но конечным) числом внутренних состояний.

9.4. Ассоциативные исчисления

Сейчас мы используем машины Тьюринга, чтобы доказать неразрешимость некоторой алгоритмической проблемы, связанной с так называемыми ассоциативными исчислениями.

Напомним, что *алфавитом* называют конечное множество, элементы его называют *символами*, или *буквами*, а конечные последовательности букв — *словами*.

Пусть фиксирован алфавит A. Будем называть *правилом* произвольную запись вида $P \to Q$, где P и Q — слова этого алфавита (мы считаем, что сама стрелка не является буквой алфавита). Будем называть *ассоциативным исчислением* конечный набор правил. (Порядок правил в наборе не играет роли.) Каждое правило рассматривается как правило преобразования слов. Именно, мы говорим, что к слову X можно применить правило $P \to Q$, если в слове X есть участок из подряд идущих букв (подслово), совпадающий с P. В этом случае его разрешается заменить на Q. Таких участков может быть несколько, так что к одному и тому же слову можно применить одно и то же правило несколькими разными способами. Кроме того, в исчислении может быть несколько правил, применимых к данному слову, и тогда можно применять любое из них. После этого можно снова применить то же самое или другое правило исчисления, и так далее.

Повторим это определение более формально. Говорят, что *слово X можно преобразовать в слово Y по правилам исчисления I*, если существует конечная последовательность слов

$$X = Z_0, Z_1, Z_2, \ldots , Z_{k-1}, Z_k = Y,$$

в которой каждое слово Z_i получается из предыдущего слова Z_{i-1} по одному из правил исчисления I, то есть в I существует такое правило $P \to Q$, что $Z_{i-1} = RPS$ и $Z_i = RQS$ для некоторых слов R и S.

Таким образом, каждому исчислению соответствует некоторое множество пар слов — множество таких пар $\langle X, Y\rangle$, что X можно преобразовать в Y по правилам этого исчисления.

Теорема 60. Для всякого исчисления это множество перечислимо. Существует исчисление, для которого это множество неразрешимо.

Мы докажем эту теорему в следующем разделе. Первая её часть доказывается легко: множество всех последовательностей $Z_0 \to Z_1 \to \ldots \to Z_k$, согласованных с правилами исчисления, разрешимо и потому перечислимо. Если от каждой из них оставить только начало и конец, получится перечисление искомого множества.

Осталось построить пример неразрешимого ассоциативного исчисления (исчисления, для которого указанное множество неразрешимо). Для этого мы покажем, что работу любой машины Тьюринга можно в некотором смысле моделировать с помощью ассоциативного исчисления, а затем возьмём исчисление, соответствующее машине с неразрешимой проблемой остановки.

9.5. Моделирование машин Тьюринга

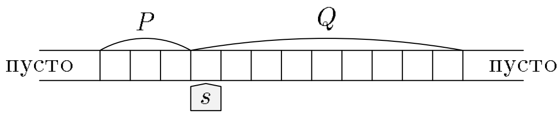
Теорема 61. Пусть M — машина Тьюринга, алфавит которой включает 0 и 1. Тогда можно построить ассоциативное исчисление I с таким свойством: двоичное слово Y является результатом работы машины на двоичном слове X тогда и только тогда, когда слово [X] по правилам исчисления I можно преобразовать в слово Y.

Напомним, что результат работы машины мы определили как максимальный блок нулей и единиц, начиная с позиции головки. Заметим также, что алфавит исчисления I содержит символы 0 и 1, дополнительные символы [и] и может содержать другие символы.

77. Покажите, что если без дополнительных символов [и] утверждение теоремы не будет верным. (Указание: если слово Y можно получить из слова X по правилам исчисления, то слово PYQ можно получить из слова PXQ по правилам исчисления.)

◁ Идея моделирования состоит в следующем. Будем кодировать конфигурацию машины Тьюринга (содержимое ленты, положение головки, состояние) в виде слова. Тогда переход от конфигурации к следующей по правилам машины Тьюринга соответствует применению правила ассоциативного исчисления.

Конечно, для этого надо правильно выбрать кодирование. Мы будем делать это так: конфигурация



кодируется словом [PsQ]. Таким образом, алфавит нашего исчисления будет включать все буквы алфавита машины Тьюринга, включая пробел (который мы изображаем как «␣»), и все её состояния (мы считаем, что множество состояний не пересекается с алфавитом), а также специальные символы [и]. Заметим, что кодирование не однозначно за счёт того, что слово P может начинаться с пробела, а слово Q им кончаться.

Например, если `a`, `b` и `c` — буквы алфавита, а `s` — состояние, то слово `[absc]` соответствует состоянию `s`, ленте `...abc...` и головке напротив `c`; слова `[_absc]` и `[absc_]` соответствуют той же конфигурации. Другие примеры кодирования конфигураций: слово `[sabc]` соответствует состоянию `s`, ленте `...abc...` и головке напротив `a`, слово `[abcs]` — ленте `...abc...` с головкой справа от `c`, а слово `[s]` соответствует пустой ленте.

Теперь надо написать правила нашего исчисления. Мы хотим, чтобы к коду любой конфигурации было применимо ровно одно правило и чтобы получающееся при его применении слово было кодом следующей конфигурации. Это можно сделать, построчно переводя таблицу переходов машины Тьюринга на язык правил. Пусть, например, в таблице есть инструкция «находясь в состоянии s и читая букву x, перейти в состояние s', напечатать букву x' и остаться на месте». Тогда мы включаем в наше исчисление правило

$$sx \to s'x'.$$

Инструкция «находясь в состоянии s и читая букву x, перейти в состояние s', напечатать букву x' и сдвинуться влево» порождает правила

$$\alpha sx \to s'\alpha x'$$

для всех букв α алфавита машины Тьюринга.

Инструкция «находясь в состоянии s и читая букву x, перейти в состояние s', напечатать букву x' и сдвинуться вправо» порождает правило

$$sx \to x's'.$$

Но надо ещё позаботиться о ситуации, когда слова P или Q пусты. Для этого нужны такие правила:

если в таблице есть инструкция ...	правило
читая x в состоянии s, перейти в состояние s', напечатать x' и сдвинуться налево	$\mathtt{[}sx \to \mathtt{[}s'\_x'$
читая пробел в состоянии s, перейти в состояние s', напечатать x' и остаться на месте	$s\mathtt{]} \to s'x'\mathtt{]}$
читая пробел в состоянии s, перейти в состояние s', напечатать x' и сдвинуться налево	$\alpha s\mathtt{]} \to s'\alpha x'\mathtt{]}$ $\mathtt{[}s\mathtt{]} \to \mathtt{[}s'\_x'\mathtt{]}$
читая пробел в состоянии s, перейти в состояние s', напечатать x' и сдвинуться направо	$s\mathtt{]} \to x's'\mathtt{]}$

Особые случаи, рассмотренные в этой таблице — это случай пробела под головкой и пустого слова Q, а также сдвига налево при пустом слове P.

Применение этих правил шаг за шагом моделирует процесс вычисления машины Тьюринга. Но надо ещё «подготовить вход» и «очистить выход». После завершения работы машина оказывается в некотором заключительном состоянии s, и код конфигурации имеет вид $\mathtt{[}PsQ\mathtt{]}$. А нам надо получить слово, являющееся результатом работы машины. То есть, по нашим правилам определения результата, надо удалить P и открывающуюся скобку, а также в Q выделить максимальное начало из нулей и единиц, и всё последующее удалить. Вот как это делается. Введём дополнительный символ $\triangleleft$, правила $s \to \triangleleft$ для каждого заключительного состояния s, правила $\alpha\triangleleft \to \triangleleft$ для всех букв α и правило $\mathtt{[}\triangleleft \to \triangleright$. Тогда символ $\triangleleft$ появится на месте заключительного состояния, съест всё слева от себя, а в конце встретит скобку и превратится в новый символ $\triangleright$. Для этого символа мы используем правила $\triangleright 0 \to 0\triangleright$, $\triangleright 1 \to 1\triangleright$ и $\triangleright\alpha \to \nabla\alpha$

(последнее правило — для всех α из алфавита машины, кроме 0 и 1, а также для $\alpha = \mathtt{]}$). Символ $\triangleright$ пропускает результат налево от себя и превращается в символ ∇. А этот символ уничтожает всё справа от себя и затем самоуничтожается в паре с закрывающейся скобкой; правила таковы: $\nabla\alpha \to \nabla$ для всех символов α из алфавита машины, а также $\nabla\mathtt{]} \to \Lambda$ (Λ обозначает пустое слово).

Эти правила позволяют выделить результат работы машины из кода заключительной конфигурации. Теперь уже можно сказать, что машина на входе X даёт результат Y тогда и только тогда, когда по этим правилам из слова $\mathtt{[}s_0X\mathtt{]}$ можно получить слово Y. Единственное различие с формулировкой теоремы состоит в том, что там нет символа s_0, но это исправить легко: добавим ещё один символ $\mathtt{[}'$, правило $\mathtt{[} \to \mathtt{[}'s_0$, и во всех остальных правилах заменим $\mathtt{[}$ на $\mathtt{[}'$.

Теперь уже всё в точности соответствует формулировке теоремы, и доказательство можно считать завершённым. (Увы, аккуратное проведение почти очевидной идеи часто требует перечисления многих деталей, в которых к тому же легко пропустить какой-нибудь случай или допустить ошибку.) $\triangleright$

Теперь уже можно построить обещанное неразрешимое ассоциативное исчисление.

Возьмём перечислимое неразрешимое множество K. Возьмём машину Тьюринга, которая на входах из K останавливается и даёт пустое слово, а на входах не из K не останавливается (полухарактеристическая функция перечислимого множества, напомним, вычислима, и потому вычислима на машине Тьюринга по тезису Тьюринга).

Построим ассоциативное исчисление, моделирующее эту машину в только что описанном смысле. Для него нет алгоритма, который по паре слов X и Y выясняет, можно ли преобразовать X в Y. В самом деле, если бы такой алгоритм был, то можно было бы применить его к словам $\mathtt{[}X\mathtt{]}$ и Λ (пустое слово) и узнать, лежит ли слово X в множестве K или не лежит.

9.6. Двусторонние исчисления

Оказывается, что рассуждение предыдущего раздела можно немного усилить. Назовём ассоциативное исчисление (т. е. набор правил) *двусторонним*, если оно вместе с каждым правилом $X \to Y$ содержит и симметричное правило $Y \to X$.

Теорема 62. Существует двустороннее исчисление, для которого нет алгоритма, выясняющего, можно ли получить одно слово из другого по правилам этого исчисления.

$\lhd$ Для доказательства мы воспользуемся той же конструкцией с небольшими изменениями. Для начала правило $\triangledown\mathtt{]} \to \Lambda$ заменим на правило $\triangledown\mathtt{]} \to \star$ (мы вскоре увидим, зачем это нужно). После этого для каждого правила добавим обратное. Это новое исчисление I' также моделирует машину Тьюринга:

Лемма. Двоичное слово Y является результатом работы машины на двоичном слове X тогда и только тогда, когда слово $\mathtt{[}\ X\ \mathtt{]}$ можно преобразовать в слово $Y\star$ по правилам исчисления I'.

Доказательство леммы. Если не добавлять обратные правила, то I' ничем не отличается от ранее построенного исчисления (кроме последнего шага, где остаётся звёздочка — но это, очевидно, несущественно). Поэтому нам надо лишь показать, что если $\mathtt{[}X\mathtt{]}$ можно преобразовать в слово $Y\star$ по правилам исчисления I', то это можно сделать без применения обратных правил, только с помощью прямых.

Доказывается это так. Назовём «активными» следующие символы алфавита исчисления I': символ $\mathtt{[}$ (который, напомним, заменяется на первом же шаге), все состояния машины, $\triangleright$, $\triangleleft$, $\triangledown$ и $\star$. Тогда в каждом правиле нашего исчисления слева и справа есть ровно один активный символ. Следовательно, в любой последовательности прямых и обратных правил, соединяющей $\mathtt{[}\ X\ \mathtt{]}$ с $Y\star$, все слова содержат по одному активному символу.

Теперь такое наблюдение: к слову, в котором один активный символ, применимо не более одного прямого правила. (Это легко проверить, посмотрев все правила; причина здесь в том, что правила моделируют работу детерминированной машины Тьюринга, в которой следующая конфигурация определена однозначно.) Поэтому можно удалить все обратные правила из последовательности преобразований `[` X `]` в $Y\star$.

В самом деле, рассмотрим последнее вхождение обратного правила в последовательность преобразований. Оно не может быть самым последним в последовательности, так как обратное правило не порождает символ $\star$. Значит, за ним следует применение какого-то прямого правила. Но одно прямое правило, которое можно применить, уже есть — это то, к которому было обратно наше обратное правило. В силу единственности другого прямого правила быть не может. Поэтому применение обратного, а за ним прямого правила можно взаимно сократить и получить более короткую последовательность, в которой снова найти последнее обратное правило и т. д. Лемма доказана.

Из неё сразу же следует существование неразрешимых двусторонних ассоциативных исчислений, которое и составляло утверждение основной теоремы этого раздела. $\triangleright$

9.7. Полугруппы, образующие и соотношения

Сейчас мы объясним, как только что доказанное утверждение о двусторонних ассоциативных исчислениях переводится на алгебраический язык. (Ничего нового по существу при этом не утверждается, это просто перевод.) Напомним некоторые сведения из алгебры.

Полугруппой называется произвольное непустое множество G с ассоциативной операцией, записываемой как умножение, причём существует единичный элемент 1, для которого $1 \times x = x \times 1 = x$ для всех $x \in G$. (Полугруппы без единичного элемента нам не потребуются.) Пусть G — некоторая полугруппа. Говорят, что множество $A \subset \subset G$ является *множеством образующих* для полугруппы G, если всякий элемент полугруппы может быть получен как произведение образующих (возможно, пустое — такое произведение считается равным 1). Полугруппа называется *конечно порождённой*, если она имеет конечное множество образующих.

Пусть $A = \{a_1, \ldots, a_n\}$ — алфавит. Тогда множество всех слов в алфавите A образует полугруппу. Умножение в ней — это *конкатенация* слов, то есть приписывание одного к другому. Единичный элемент — пустое слово. Очевидно, $a_1, \ldots, a_n$ являются образующими этой полугруппы. (Точнее следовало бы говорить об однобуквенных словах как образующих.) Эта полугруппа называется *свободной полугруппой* с образующими $a_1, \ldots, a_n$. Будем обозначать её $\mathcal{F}(a_1, \ldots, a_n)$.

Пусть G — произвольная полугруппа, и $g_1, \ldots, g_n$ — любые её элементы. Тогда существует единственный гомоморфизм h полугруппы $\mathcal{F}(a_1, \ldots, a_n)$ в полугруппу G, для которого $h(a_i) = g_i$. (*Гомоморфизмом* полугрупп называют отображение, при котором образ произведения равен произведению образов и единица переходит в единицу.) Он переводит слово из символов a_i в произведение соответствующих элементов g_i; образом пустого слова является единичный элемент. Очевидно, образ этого гомоморфизма совпадает со всей полугруппой G тогда и только тогда, когда элементы $g_1, \ldots, g_n$ являются образующими группы G.

Соотношениями мы будем называть равенства вида $X = Y$, где X и Y — элементы свободной полугруппы $\mathcal{F}(a_1, \ldots, a_n)$, то есть слова в алфавите A. Говорят, что соотношение $X = Y$ выполнено в полугруппе G с выделенными элементами $g_1, \ldots, g_n$, если образы слов X и Y при описанном гомоморфизме, то есть произведения соответствующих элементов g_i, равны. Пусть имеется некоторый набор соотношений $X_1 = Y_1, \ldots, X_k = Y_k$. Будем рассматривать различные полугруппы G с выделенными n элементами, в которых выполнены все эти соотношения, и в которых выделенные элементы являются образующими. (Например, полугруппа из единственного единичного элемента также входит в их число.) Среди таких полугрупп, как мы сейчас увидим, есть «максимальная», в которой выполнено меньше всего соотношений.

Введём на словах алфавита A отношение эквивалентности, считая, что $P \equiv Q$, если P можно преобразовать в Q в двустороннем ассоциативном исчислении с правилами $X_1 \leftrightarrow Y_1, \ldots, X_k \leftrightarrow Y_k$. (Другими словами, в любом слове разрешается заменить его подслово X_i на подслово Y_i и наоборот.) Очевидно, это отношение действительно будет отношением эквивалентности. Заметим, что если $P \equiv Q$, то $PR \equiv QR$ и $RP \equiv RQ$ для произвольного слова R (в цепочке преобразований можно дописать ко всем словам R слева или справа). Рассмотрим классы эквивалентности этого отношения. На них можно определить операцию произведения, считая произведением двух классов, содержащих слова P и Q, класс, содержащий их конкатенацию PQ. Отмеченное свойство отношения эквивалентности гарантирует корректность этого определения (класс-произведение не зависит от выбора представителей в сомножителях). Тем самым мы получаем полугруппу G, в которой единичным элементом будет класс пустого слова, а классы $g_i = [a_i]$ эквивалентности однобуквенных слов будут образующими. Её обозначают

$$\mathcal{F}(a_1, \ldots, a_n)/(X_1 = Y_1, \ldots, X_k = Y_k)$$

и называют *полугруппой с образующими* $a_1, \ldots, a_k$ *и соотношениями* $X_1 = Y_1, \ldots, X_k = Y_k$. Ясно, что в полугруппе G выполнены исходные соотношения $X_i = Y_i$. Легко понять также, что в ней нет никаких других соотношений, кроме следствий исходных:

Теорема 63. Если соотношение $X = Y$ выполнено в полугруппе

$$\mathcal{F}(a_1, \ldots, a_n)/(X_1 = Y_1, \ldots, X_k = Y_k),$$

то оно выполнено в любой полугруппе G с выделенными элементами $g_1, \ldots, g_n$, в которой выполнены все соотношения $X_i = Y_i$.

$\lhd$ В самом деле, словам X и Y соответствуют их классы эквивалентности по описанному отношению, так что из X можно получить Y по правилам двустороннего ассоциативного исчисления. Но все эти правила не меняют значение элемента в любой полугруппе, в которой выполнены соотношения $X_i = Y_i$, и потому в любой такой полугруппе словам X и Y будет соответствовать один и тот же элемент. $\triangleright$

78. Рассмотрим полугруппу с двумя образующими a_1 и a_2 и соотношениями $a_1a_2 = \Lambda$, $a_2a_1 = \Lambda$. Что это за полугруппа?

79. Рассмотрим полугруппу с двумя образующими a_1 и a_2 и соотношением $a_1a_2 = a_2a_1$. Что это за полугруппа?

80. Рассмотрим полугруппу с двумя образующими a_1 и a_2 и соотношениями $a_1a_1 = \Lambda$, $a_2a_2 = \Lambda$, $a_1a_2 = a_2a_1$. Что это за полугруппа?

81. Рассмотрим полугруппу с двумя образующими a_1 и a_2 и соотношениями $a_1a_1 = \Lambda$, $a_2a_2a_2 = \Lambda$, $a_1a_2 = a_2a_2a_1$. Что это за полугруппа?

Теперь мы можем сформулировать утверждение о существовании неразрешимых двусторонних ассоциативных исчислений в терминах полугрупп.

Теорема 64. Существует полугруппа, с конечным числом образующих и конечным числом соотношений, в которой задача проверки равенства двух слов, составленных из образующих, алгоритмически неразрешима (нет алгоритма, который по двум словам такого вида узнавал, равны они в этой полугруппе или не равны).

$\triangleleft$ Согласно определению, равенство двух слов, составленных из образующих, означает возможность преобразовать одно из них в другое по правилам двустороннего исчисления, так что эта теорема представляет собой переформулировку теоремы 62 (с. 124) $\triangleright$

Эта теорема была доказана в 1947 году независимо Постом и Андреем Андреевичем Марковым (младшим); вскоре после этого Пётр Сергеевич Новиков усилил её, построив пример группы (а не только полугруппы!) с конечным числом образующих и соотношений, для которой проблема равенства двух слов из образующих неразрешима.

10. Арифметичность вычислимых функций

10.1. Программы с конечным числом переменных

Мы хотим показать, что график всякой вычислимой функции является арифметическим множеством, то есть выразим формулой арифметики. Для этого удобно перейти от машин Тьюринга к другой модели, которую можно условно назвать машинами с конечным числом регистров.

Программа для такой машины использует конечное число переменных, значениями которых являются натуральные числа. Числа эти могут быть произвольного размера, так что машина реально имеет память неограниченного объёма. Программа состоит из нумерованных по порядку команд. Каждая команда имеет один из следующих видов:

- `a:=0`
- `a:=b`
- `a:=b+1`
- `a:=b-1`
- `goto` ⟨номер⟩
- `if a=0 then goto` ⟨номер1⟩ `else goto` ⟨номер2⟩
- `stop`

Для тех, кто не сталкивался с командой `goto`, поясним, что имеется в виду в команде `if`: если значение переменной `a` равно нулю, то далее исполняется команда с номером, указанным после `then` (этот номер — конкретное натуральное число, не превосходящее числа команд в программе); если `a` не равно нулю, то дальше выполняется команда с номером, указанным после `else`. Команда `goto` без условия всегда выполняет переход к команде с указанным в ней номером.

Поскольку мы считаем, что значения переменных — натуральные (целые неотрицательные) числа, условимся считать разность $0 - 1$ равной 0 (впрочем, это не так важно — можно было бы считать это аварией).

Дойдя до команды `stop`, программа заканчивает работу.

Как и для машин Тьюринга, полезна некоторая практика программирования. Для тренировки напишем программу сложения двух чисел. Она помещает в `c` сумму чисел, которые были в переменных `a` и `b`. Такая программа на паскале имела бы вид

```
c:=a;
{инвариант: ответ=сумма текущих значений c и b}
while b<>0 do begin
   c:=c+1;
   b:=b-1;
end;
```

Имитируя цикл с помощью операторов перехода, получаем программу для нашей машины:

```
1 c:=a
2 if b=0 then goto 6 else goto 3
3 c:=c+1
4 b:=b-1
5 goto 2
6 stop
```

Теперь легко понять, как написать программы для вычитания, умножения (которое реализуется как цикл с повторным сложением), деления с остатком (как учил ещё Энгельс в забытой ныне книге «Диалектика природы», деление есть сокращённое вычитание), возведения в степень, проверки простоты, отыскания n-го простого числа и т. п. Вообще по сравнению с машинами Тьюринга этот язык более привычен и потому легче поверить, что на нём можно запрограммировать все алгоритмы.

Единственное, чего в нём реально не хватает — это массивов. Но это легко обойти, поскольку есть числа произвольного размера и нас не интересует число операций (как это принято в общей теории алгоритмов). Вместо массива битов мы можем хранить число, двоичной записью которого он является, а для массивов чисел воспользоваться, скажем, основной теоремой арифметики и хранить последовательность $\langle a, b, c, d, e\rangle$ как число $2^a 3^b 5^c 7^d 11^e$. При этом операции `a[i]:=b` и `b:=a[i]` заменяются на небольшие программы, которые содержат переменные `a`, `b`, `i` и ещё несколько переменных. (Частью этих программ является нахождение простого числа с заданным порядковым номером.)

Легко определить понятие вычислимой (в этой модели) функции. Пусть есть программа с двумя переменными `x` и `y` (и, возможно, другими). Поместим в `x` некоторое число n, а в остальные переменные поместим нули. Запустим программу. Если она не остановится, то вычисляемая ей функция в точке n не определена. Если остановится, то содержимое переменной `y` после остановки и будет значением функции, вычисляемой нашей программой (в точке n). Функция называется вычислимой (в этой модели), если существует вычисляющая её программа.

Как всегда, при определении мы фиксировали различные детали, большинство из которых не являются существенными. Можно было бы добавить некоторые команды (сложение, например) — или даже исключить (например, без копирования можно обойтись с помощью небольшой хитрости).

82. Покажите, что класс вычислимых функций не изменится, если исключить из определения команду копирования `a:=b`.

Несколько более удивительно, что число переменных можно ограничить, скажем, сотней — но и это не так уж странно, если вспомнить, что мы можем хранить в одной переменной целый массив.

83. Проверьте это.

10.2. Машины Тьюринга и программы

Построенная вычислительная модель не слабее машин Тьюринга в том смысле, что любую вычислимую на машинах Тьюринга функцию можно вычислить и программой с конечным числом переменных.

Теорема 65. Всякая функция, вычислимая на машинах Тьюринга, может быть вычислена с помощью программы описанного вида с конечным числом переменных.

Следует уточнить, однако, что мы имеем в виду, так как для машин Тьюринга исходное данное и результат были двоичными словами, а для программ — натуральными числами. Мы отождествляем те и другие по естественному правилу, при котором слова Λ (пустое), 0, 1, 00, 01, ... соответствуют числам 0, 1, 2, 3, 4 ... (чтобы получить из числа слово, прибавим к нему единицу, переведём в двоичную систему и отбросим единицу в старшем разряде).

$\triangleleft$ Как и раньше, мы приведём лишь приблизительное описание того, как по машине Тьюринга строится программа с конечным числом переменных, вычисляющая ту же функцию. Прежде всего конфигурации машины Тьюринга надо закодировать числами. Можно сделать это, например, поставив в соответствие каждой конфигурации четыре числа: номер текущего состояния, номер текущего символа (в ячейке, где стоит головка машины), код содержимого ленты слева от головки и код содержимого ленты справа от головки.

Чтобы решить, как удобнее кодировать содержимое ленты слева и справа от головки, заметим, что машина Тьюринга обращается с двумя половинами лентами слева и справа от головки как со стеками. (Стеком называется структура данных, напоминающая стопку листов. В неё можно положить лист наверх, взять верхний лист, а также проверить, есть ли ещё листы.) В самом деле, при движении головки направо из правого стека берётся верхний элемент, а в левый стек кладётся; при движении налево — наоборот. Стеки легко моделировать с помощью чисел: например, если в стеке хранятся символы 0 и 1, то добавление нуля соответствует операции $x \mapsto 2x$, добавление единицы — операции $x \mapsto 2x+1$, верхний элемент есть остаток при делении на 2, а удаление верхнего элемента есть деление на 2 (с отбрасыванием остатка). Другими словами, мы воспринимаем двоичную запись числа как стек, вершина которого находится справа, у младшего разряда. Точно так же можно использовать n-ичную систему счисления и представить стек с n возможными символами в каждой позиции.

Теперь основной цикл машины Тьюринга можно записать как программу, оперирующую с указанными четырьмя числами (символ, состояние, левый стек и правый стек) — без особых хитростей. Но несколько вещей всё-таки надо иметь в виду.

Во-первых, стеки конечны, а лента бесконечна — мы должны договориться, что если стек опустошается, то в него автоматически добавляется символ пробела. Тем самым бесконечный пустой хвост ленты может присутствовать в стеке, как теперь говорят, виртуально.

Во-вторых, напомним, что мы договорились отождествлять двоичные слова (которые подаются на вход машины Тьюринга) и их коды (которые хранятся в переменных нашей программы). Поэтому, получив код входного слова, надо его разобрать по символам и положить эти символы один за другим в стек (основания систем счисления разные, так что просто так переписать это нельзя). Аналогичные проблемы возникают и при превращении выхода (части содержимого правого стека) в соответствующее число, но все они легко преодолимы, и мы не будем вдаваться в подробности. $\triangleright$

Верно и обратное утверждение:

Теорема 66. Всякая функция, вычислимая программой с конечным числом переменных, вычислима на машине Тьюринга.

$\triangleleft$ Нам надо моделировать поведение программы с помощью машины Тьюринга. Будем считать, что значения переменных записаны на ленте (в двоичной системе) и разделены специальным разделительным символом. Тогда машина может найти любую переменную, идя от начала ленты и считая разделительные символы, сделать что-то с этой переменной и затем вернуться обратно в начало. (Нет необходимости записывать на ленте номер исполняемой команды, поскольку команд конечное число и машина может помнить номер текущей команды как часть своего состояния.) Операции прибавления и вычитания единицы также легко выполнимы в двоичной записи (если идти справа налево). Надо только иметь в виду, что размер числа может увеличиться, и тогда нужно для него освободить место, сдвинув все символы справа от головки на одну позицию. (При уменьшении нужно сдвинуть влево.) Ясно, что это также легко выполнить с помощью машины Тьюринга.

Если мы записываем числа в двоичной системе, то проблемы с перекодированием при вводе-выводе минимальны (надо лишь дописать нули для значений остальных переменных в начале работы и встать в нужное место ленты в конце). $\triangleright$

10.3. Арифметичность вычислимых функций

Сейчас мы докажем, что функции, вычислимые программами с конечным числом переменных, арифметичны, то есть их графики являются арифметическими множествами. В этом разделе мы вновь предполагаем некоторое знакомство читателя с логическими обозначениями, и будем рассматривать *арифметические формулы*, содержащие переменные по натуральным числам, равенство, константы 0 и 1, операции сложения и умножения, логические связки (И, ИЛИ, НЕ) и кванторы «для всех» и «существует». Формально говоря, мы рассматриваем сигнатуру, содержащую единственный двуместный предикатный символ (равенство), две константы 0 и 1 и два двуместных функциональных символа (сложение и умножение). Говоря об истинности таких формул, мы имеем в виду их истинность в стандартной интерпретации, носителем которой является множество $\mathbb{N}$ натуральных чисел.

Множество $A \subset \mathbb{N}^k$ называется *арифметическим*, если существует арифметическая формула α с параметрами $x_1, \ldots, x_k$, которая его представляет в следующем смысле: $\langle n_1, \ldots, n_k \rangle \in A$ тогда и только тогда, когда формула α истинна при значениях параметров $x_1 = n_1$, $\ldots, x_k = n_k$.

Теорема 67. График любой функции, вычисляемой программой с конечным числом переменных, является арифметическим множеством.

$\lhd$ Пусть $f\colon \mathbb{N} \to \mathbb{N}$ — функция, вычислимая некоторой программой P с конечным числом переменных $k_1, \ldots, k_N$. Будем считать, что входной переменной является k_1, а выходной — k_2. Нам нужно написать формулу с двумя переменными x, y, которая была бы истинна тогда и только тогда, когда $y = f(x)$. Состояние программы с конечным числом переменных полностью описывается значениями переменных и номером текущей команды (в процессорах соответствующий регистр часто называют program counter, по-русски счётчик команд). Легко понять, что соответствие между двумя последовательными состояниями программы с конечным числом переменных арифметично. Мы имеем в виду, что можно написать арифметическую формулу

$$\mathrm{Step}(s_1, \ldots, s_N, p, s'_1, \ldots, s'_N, p'),$$

с $2N + 2$ переменными, которая утверждает, что данная программа P из состояния, где переменные равны $s_1, \ldots, s_N$, а счётчик команд равен p, за один шаг переходит в состояние, где переменные равны $s'_1, \ldots, s'_n$, а счётчик команд равен p'. (Договоримся, что значение $p' = 0$ соответствует остановке программы.) Такая формула является конъюнкцией отдельных утверждений, соответствующих каждой строке программы. Пусть, например, строка 7 программы имеет вид `k₂:=k₃`. Тогда в конъюнкции будет член вида

$$(p = 7) \Rightarrow ((s'_1 = s_1) \wedge (s'_2 = s_3) \wedge (s'_3 = s_3) \wedge \ldots \\ \ldots \wedge (s'_N = s_N) \wedge (p' = 8)).$$

Для строки с условными переходами типа

```
3  if k₅=0 then goto 17 else goto 33
```

в формуле будет два конъюнктивных члена (на два случая перехода)

$$((p=3)\wedge(s_5=0))\Rightarrow((s'_1=s_1)\wedge\ldots\wedge(s'_N=s_N)\wedge(p'=17))$$

и

$$((p=3)\wedge(s_5\neq 0))\Rightarrow((s'_1=s_1)\wedge\ldots\wedge(s'_N=s_N)\wedge(p'=33)).$$

Надо ещё добавить утверждение о том, что при $p=0$ работа прекращается, то есть что переменные на следующем шаге сохраняют свои значения, и p' остаётся равным 0.

Таким образом, арифметичность одного шага работы программы доказать несложно. Остаётся главный вопрос: как записать в виде формулы тот факт, что существует *последовательность* шагов, которая начинается с исходного состояния, заканчивается в данном и в которой каждый шаг правилен. Трудность в том, что здесь нужно как бы написать переменное число кванторов существования — или квантор «существует конечная последовательность натуральных чисел».

Это делается с помощью приёма, традиционно называемого β-функцией Гёделя. Вот что имеется в виду.

Лемма 1. Для любого k можно найти сколь угодно большое целое положительное число b, при котором первые k членов последовательности $b+1, 2b+1, 3b+1, \ldots$ попарно взаимно просты.

Доказательство. Любой общий простой делитель двух из этих чисел будет делителем их разности, то есть числа lb при $0<l<k$; взяв b кратным $k!$, мы гарантируем, что он будет делителем числа b, но все члены нашей последовательности взаимно просты с b. Лемма доказана.

Лемма 2. Для любой последовательности $x_0, x_1, \ldots, x_n$ натуральных чисел можно найти такие числа a и b, что x_i есть остаток от деления a на $b(i+1)+1$.

Доказательство. Согласно предыдущей лемме, делители $b(i+1)+1$ можно взять взаимно простыми (и сколь угодно большими), остаётся воспользоваться «китайской теоремой об остатках». Эта теорема утверждает, что если целые положительные числа $d_1, \ldots, d_k$ взаимно просты, то при делении целого u на них может получиться любой заданный набор остатков. В самом деле, таких наборов будет $d_1 d_2 \ldots d_k$ (поскольку при делении на d_i возможны остатки от 0 до $d_i - 1$). При делении чисел $u = 0, 1, \ldots, d_1 d_2 \ldots d_k - 1$ получаются разные наборы остатков (если два числа u' и u'' дают одинаковые остатки, то их разность делится на все d_i, что невозможно в силу взаимной простоты). Поэтому чисел столько же, сколько наборов остатков, и должны появиться все наборы. Лемма 2 доказана.

Эта лемма показывает, что последовательность произвольной длины можно закодировать тремя числами a, b и n (последнее число — длина последовательности). Таким образом, условно говоря, можно заменить «формулу»

$$\exists \langle x_0, \ldots, x_n \rangle (\forall i \leqslant n)[\ldots x_i \ldots]$$

(которая на самом деле не является арифметической формулой, так как содержит квантор по конечным последовательностям) на формулу

$$\exists a \, \exists b \, \exists n \, (\forall i \leqslant n)[\ldots (\text{остаток от деления } a \text{ на } b(i+1)+1) \ldots].$$

Мы будем записывать остаток от деления a на $b(i+1)+1$ как $\beta(a, b, i)$ (отсюда и название «бета-функция»).

Возвращаясь к нашей программе P с конечным числом переменных $k_1, \ldots, k_N$ и вычисляемой ей функции f, можно записать утверждение вида $f(x) = y$ так: существуют такое число шагов n и такие числа $a_1, b_1, a_2, b_2, \ldots, a_N, b_N, a, b$, что

- $\beta(a_1, b_1, 0), \ldots, \beta(a_N, b_N, 0)$ есть правильные начальные значения переменных (первое равно x, остальные равны 0); $\beta(a, b, 0)$ есть правильное начальное значения счётчика команд, то есть 1;

- для каждого i от 0 до $n-1$ имеет место

$$\mathrm{Step}(\beta(a_1,b_1,i),\ldots,\beta(a_N,b_N,i),\beta(a,b,i),\\ \beta(a_1,b_1,i+1),\ldots,\beta(a_N,b_N,i+1),\beta(a,b,i+1)),$$

  то есть каждый переход соответствует программе;

- $\beta(a_2,b_2,n)=y$ (значение выходной переменной k_2 в конце вычисления равно y) и $\beta(a,b,n)=0$ (значение счётчика команд в конце вычисления равно 0, что по нашей договорённости соответствует остановке машины).

Итак, арифметичность вычислимых (на машинах с конечным числом переменных) функций доказана. $\triangleright$

Вспоминая теорему 65, мы заключаем, что всякая вычислимая на машине Тьюринга функция арифметична. Принимая тезис Тьюринга, можно сказать, что график любой вычислимой функции является арифметическим множеством.

10.4. Теоремы Тарского и Гёделя

Поскольку графики вычислимых функций арифметичны, очевидно, разрешимые и перечислимые множества тоже будут арифметическими. Тем самым мы можем оправдать название «арифметическая иерархия» для классов Σ_n и Π_n:

Теорема 68. При любом n любое множество из класса Σ_n или Π_n арифметично (есть формула арифметики, выражающая принадлежность этому множеству).

$\triangleleft$ После того, как мы доказали арифметичность вычислимых функций, всё ясно — множества из классов Σ_n и Π_n получаются навешиванием кванторов на разрешимые множества, а разрешимое множество арифметично. $\triangleright$

Верно и обратное:

Теорема 69. Всякое арифметическое множество лежит в классе Σ_n или Π_n для некоторого n (и, естественно, для всех бóльших n).

◁ Формулу, задающую арифметическое множество, приведём к предварённой нормальной форме (вынеся кванторы наружу). Ясно, что бескванторная часть задаёт разрешимое множество, поэтому исходное множество принадлежит какому-то из классов Σ_n или Π_n.

Можно и не использовать предварённой нормальной формы, а применить индукцию по длине формулы и сослаться на то, что пересечение, объединение и дополнение, а также проекция не выводят за пределы арифметической иерархии (объединения всех классов Σ_n и Π_n). ▷

Рассмотрим теперь множество T, элементами которого являются все истинные арифметические формулы без параметров (точнее, их номера в какой-то вычислимой нумерации всех формул.)

Теорема 70. Любое арифметическое множество m-сводится к множеству T.

◁ Это утверждение почти очевидно. Пусть A — произвольное арифметическое множество. Пусть $\alpha(x)$ — формула с одной переменной, которая выражает принадлежность множеству A. Это означает, что $\alpha(n)$ истинно при тех и только тех n, которые принадлежат A. Тогда вычислимая функция $n \mapsto$ (номер формулы, которая является результатом подстановки константы n в $\alpha(x)$) m-сводит A к T. ▷

Теорема 71. Множество T не арифметично.

◁ После проведённой нами подготовки это утверждение очевидно: если бы T было арифметическим, то оно лежало бы в некотором конкретном классе Σ_n. Поскольку всякое арифметическое множество сводится к T, то по теореме 53 все арифметические множества лежали бы в этом классе. Но мы знаем, что множества из более высоких классов иерархии тоже арифметичны, но в Σ_n не лежат. ▷

Эта теорема называется *теоремой Тарского*. Её можно прочесть так: множество арифметических истин не арифметично. Или: понятие арифметической истины невыразимо в арифметике.

Теорема 72. Множество T арифметических истин неперечислимо.

◁ В самом деле, любое перечислимое множество арифметично. ▷

Это утверждение называется *теоремой Гёделя о неполноте*. Его можно переформулировать так: всякое исчисление, порождающее формулы арифметики (т. е. алгоритм, перечисляющий некоторое множество таких формул) либо *неадекватно* (порождает некоторую ложную формулу), либо *неполно* (не порождает некоторой истинной формулы).

84. Покажите, что для любого N множество всех истинных замкнутых арифметических формул, содержащих не более N кванторов, арифметично.

85. Сформулируйте и докажите аналогичное утверждения для формул ограниченной кванторной глубины (число вложенных кванторов) и для формул с ограниченным числом перемен кванторов в предварённой нормальной форме.

10.5. Прямое доказательство теорем Тарского и Гёделя

В нашем изложении теоремы Тарского и Гёделя получились как простые следствия определений и фактов, связанных с теорией алгоритмов. С одной стороны, это позволяет лучше понять место этих теорем в общем контексте математической логики и теории алгоритмов. С другой стороны, возникает естественное желание развернуть цепочку и получить более прямые доказательства. Таковые мы сейчас и приведём.

Начиная доказывать теорему Тарского, предположим, что множество номеров всех истинных арифметических формул (без параметров) арифметично. Пусть T — это множество, а $\tau(x)$ — соответствующая формула. Перенумеруем также все формулы с одним параметром x; пусть $F_n(x)$ — формула, имеющая номер n в этой нумерации. Рассмотрим формулу с единственным параметром x, утверждающую, что результат подстановки константы x в x-ую формулу с параметром ложен. Эту формулу можно написать так:

$$\exists z(\neg\tau(z) \wedge \mathrm{Subst}(z, x, x)),$$

где $\mathrm{Subst}(p, q, r)$ — формула с тремя параметрами, выражающая такое свойство: «p есть номер (в нумерации всех формул без параметров) той формулы, которая получится, если в q-ю формулу с одним параметром подставить константу r вместо этого параметра». Записанное в кавычках свойство описывает график некоторой вычислимой функции (соответствующей простым синтаксическим действиям и переходу от одной нумерации к другой) и потому существует выражающая его формула.

Итак, мы написали некоторую формулу с единственным параметром x. Пусть она имеет некоторый номер N. Подставим этот номер N вместо её параметра. Получится некоторая формула без параметров. Из построения видно, что эта формула истинна тогда и только тогда, когда результат подстановки числа N в формулу номер N (то есть сама эта формула!) ложен.

Полученное противоречие завершает прямое доказательство теоремы Тарского. Мы видим, что нам потребовалась выразимость в арифметике не любых вычислимых функций, а одной вполне конкретной. При достаточном терпении соответствующую формулу можно-таки написать, и тем самым доказательство станет совсем «осязаемым».

Теперь изложим в том же стиле доказательство теоремы Гёделя. Как мы уже говорили, исчисление — это механизм (алгоритм), который позволяет порождать некоторые формулы языка арифметики (для простоты будем считать, что порождаются только формулы без параметров). Таким образом, возникает некоторое перечислимое множество, которое обычно задают как проекцию разрешимого множества. Именно, вводят некоторое понятие *доказательства*. При этом доказательства являются словами в некотором алфавите. Множество доказательств разрешимо, то есть есть алгоритм, отличающий настоящие доказательства от текстов, который таковыми не являются. Кроме того, есть (также разрешимое) свойство двух слов x и y, которое гласит, что x есть доказательство формулы y. Перенумеровав все доказательства и формулы и выразив указанные разрешимые свойства в языке арифметики, мы приходим к формуле $\mathrm{Proof}(x, y)$, которая истинна, когда x есть номер доказательства формулы с номером y.

Теперь напишем формулу с одним параметром x, которая говорит, что результат подстановки числа x вместо параметра в x-ую формулу с одним параметром не имеет доказательства:

$$\neg\exists z\exists p[\mathrm{Subst}(z, x, x) \wedge \mathrm{Proof}(p, z)]$$

Эта формула имеет единственный параметр (x); пусть её номер в нумерации таких формул равен N. Подставим N вместо параметра. Получится формула без параметров φ. По построению формула φ истинна, когда результат подстановки N в N-ю формулу с одним параметром недоказуем. А этот результат есть сама формула φ, так что она истинна тогда и только тогда, когда недоказуема. Значит, наше исчисление либо позволяет доказать ложную формулу φ (если φ ложна; в таком случае его называют неадекватным), либо не позволяет доказать истинную формулу φ.

Заметим, что оба этих доказательства напоминают как построение неподвижной точки, так и классический парадокс лжеца:

$$\boxed{\text{УТВЕРЖДЕНИЕ В РАМКЕ ЛОЖНО}}$$

10.6. Арифметическая иерархия и число перемен кванторов

Мы установили, что класс арифметических множеств есть объединение всех классов Σ_n и Π_n. Посмотрим подробнее, как связан номер класса с числом перемен кванторов в арифметической формуле.

Чтобы сделать это, надо уточнить, какие формулы мы считаем бескванторными. Если представлять сложение и умножение предикатными символами, то бескванторные формулы будут логическими комбинациями формул $x_i + x_j = x_k$ и $x_i \times x_j = x_k$; при этом уже для записи формулы $a + b + c = d$ нам понадобится квантор: она записывается как $\exists u((a + b = u) \wedge (u + c = d))$. Мы же, как это обычно делается, будем считать сложение и умножение функциональными символами и тем самым разрешим использовать произвольные многочлены с целыми коэффициентами в бескванторных формулах. При этом бескванторными формулами будут логические комбинации выражений вида $P = Q$, где P и Q — произвольные многочлены с целыми коэффициентами. Тогда верна такая теорема:

Теорема 73. Множество $A \subset \mathbb{N}$ принадлежит классу Σ_n $(n \geqslant 1)$ тогда и только тогда, когда принадлежность ему выражается формулой в предварённой нормальной форме, начинающейся с квантора существования и содержащей n групп одноимённых кванторов.

(Переходя к дополнению, видим, что множество принадлежит классу Π_n тогда и только тогда, когда принадлежность ему выражается в предварённой нормальной форме с n группами одноимённых кванторов, начинающейся с квантора всеобщности.)

$\triangleleft$ Мы докажем эту теорему лишь отчасти. Во-первых, если множество задаётся формулой в предварённой нормальной форме с n группами кванторов, то оно принадлежит классу Σ_n (или Π_n — в зависимости от того, какой квантор первый) — напомним, что k подряд идущих одноимённых кванторов можно заменить на один с помощью вычислимой биекции между $\mathbb{N}^k$ и $\mathbb{N}$.

Более сложно доказать обратное утверждение. Основную трудность представляет случай $n = 1$. В 1970 году Ю. В. Матиясевич, решая 10-ю проблему Гильберта, показал, что всякое перечислимое множество есть множество неотрицательных значений многочлена от нескольких переменных с целыми коэффициентами при натуральных значениях переменных. А это свойство, очевидно, записывается в виде формулы с кванторной приставкой из кванторов существования. Тем самым случай $n = 1$ был разобран.

Если считать результат Матиясевича известным, то доказательство заканчивается, поскольку каждый следующий класс получается из предыдущего навешиванием очередного квантора.

Посмотрим, однако, что можно извлечь из приведённого нами доказательства арифметичности вычислимых функций. Пусть дано произвольное перечислимое множество. Его можно представить как область определения вычислимой функции, все значения которой равны нулю, и применить описанную нами процедуру. Тогда получится формула, начинающаяся с кванторов существования (существуют числа, кодирующие последовательности значений каждой переменной в смысле β-кодирования по Гёделю). Затем идёт квантор всеобщности (по номеру шага работы программы: на каждом шаге переход должен соответствовать программе). Затем идёт формула, которая была бы бескванторной, если бы не содержала в себе остатков от деления одного выражения на другое. Но их можно устранить, добавив ещё кванторы всеобщности: утверждение типа

$$\forall i\,[\ldots\text{остаток от деления } P \text{ на } Q\ldots],$$

где P и Q — некоторые выражения, содержащие переменные (из определения β-кодирования) можно заменить по определению остатка на

$$\forall i\,\forall u\,\forall v\,[(P = uQ + v)\wedge(v < Q)] \Rightarrow [\ldots v\ldots].$$

Следовательно, мы можем представить любое перечислимое множество в виде $\exists\ldots\exists\forall\ldots\forall$-формулы. Тем самым любое множество из класса Σ_n можно представить формулой в предварённой нормальной форме с $n+1$ группами кванторов, начинающейся с кванторов существования, а любое множество из класса Π_n можно представить формулой с $n+1$ группами кванторов, начинающейся с квантора всеобщности. $\triangleright$

11. Рекурсивные функции

11.1. Примитивно рекурсивные функции

Программы с конечным числом переменных напоминали ассемблер; рассматриваемые в этом разделе рекурсивные функции скорее напоминают функциональное программирование, когда одни функции определяются через другие. Мы будем рассматривать функции с натуральными аргументами и значениями. Вообще говоря, функции могут быть не всюду определенными, так что говоря о функции n аргументов (функции из $\mathbb{N}^n$ в $\mathbb{N}$, n-местной функции), мы имеем в виду функцию, определённую на некотором подмножестве $\mathbb{N}^n$ со значениями в $\mathbb{N}$.

Пусть имеется k-местная функция f и k штук n-местных функций $g_1, \ldots, g_k$. Тогда из них можно сформировать одну n-местную функцию

$$\langle x_1, \ldots, x_n \rangle \mapsto f(g_1(x_1, \ldots, x_n), \ldots, g_k(x_1, \ldots, x_n)).$$

Говорят, что определённая таким образом функция получена из функций f и $g_1, \ldots, g_k$ с помощью операции *подстановки*.

Другая операция, называемая операцией *рекурсии*, или *примитивной рекурсии*, применяется к k-местной функции f и $(k+2)$-местной функции g. Её результатом будет $(k+1)$-местная функция h, определяемая так:

$$\begin{aligned} h(x_1, \ldots, x_k, 0) &= f(x_1, \ldots, x_k); \\ h(x_1, \ldots, x_k, y+1) &= g(x_1, \ldots, x_k, y, h(x_1, \ldots, x_k, y)). \end{aligned}$$

В последовательности $h(x_1, \ldots, x_n, 0), h(x_1, \ldots, x_n, 1), \ldots$ каждое значение определяется через предыдущее, поэтому если какое-то из значений не определено, то не определены и все последующие.

Для единообразия будем считать, что нуль-местные функции (функции без аргументов) суть константы; это позволяет рекурсивно определять функции одной переменной.

Примитивно рекурсивными называют функции, которые можно получить с помощью операций подстановки и рекурсии из следующих *базисных* функций: константы 0, операции прибавления единицы $s\colon x \mapsto x+1$ и семейства функций проекции: это семейство для каждого k содержит k штук k-местных функций $\pi_k^i(x_1, \ldots, x_k) = x_i$.

Функции проекции позволяют выполнять «неоднородные» подстановки: скажем, можно получить функцию $\langle x, y\rangle \mapsto f(g(x), h(y, x, y), x)$ из функций f и h, комбинируя их с функциями проекции: сначала получаем функцию $\langle x, y\rangle \mapsto g(x)$ (подстановка π_2^1 в g), затем $\langle x, y\rangle \mapsto h(y, x, y)$ (подстановка $\pi_2^2, \pi_2^1, \pi_2^2$ в h), затем полученные две функции вместе с функцией π_2^1 подставляем в f.

Подставляя константу 0 в функцию прибавления единицы, получаем константу (функцию нуля аргументов) 1. Затем можно получить константы 2, 3 и т. д.

11.2. Примеры примитивно рекурсивных функций

Как и с другими вычислительными моделями, важно накопить некоторый программистский опыт.

Сложение. Функция $\langle x, y\rangle \mapsto \mathrm{sum}(x, y) = x + y$ получается с помощью рекурсии:

$$\begin{aligned} \mathrm{sum}(x, 0) &= x; \\ \mathrm{sum}(x, y+1) &= \mathrm{sum}(x, y) + 1. \end{aligned}$$

Надо, конечно, представить правую часть второго равенства как результат подстановки. Формально говоря, $h(x, y, z)$ в определении рекурсии надо положить равным $s(z)$, где s — функция прибавления единицы.

Умножение. Функция $\langle x, y\rangle \mapsto \mathrm{prod}(x, y) = xy$ получается с помощью рекурсии (с использованием сложения):

$$\begin{aligned} \mathrm{prod}(x, 0) &= 0; \\ \mathrm{prod}(x, y+1) &= \mathrm{prod}(x, y) + x. \end{aligned}$$

Аналогичным образом можно перейти от умножения к возведению в степень.

Усечённое вычитание. Мы говорим об «усечённом вычитании» $x \dot{-} y = x - y$ при $x \geqslant y$ и $x \dot{-} y = 0$ при $x < y$, поскольку мы имеем дело только с натуральными (целыми неотрицательными) числами. Одноместная функция усечённого вычитания единицы определяется рекурсивно:

$$0 \dot{-} 1 = 0;$$
$$(y+1) \dot{-} 1 = y.$$

(Рекурсия здесь формальна, так как предыдущее значение не используется.) После этого усечённое вычитание для произвольных аргументов можно определить так:

$$x \dot{-} 0 = x;$$
$$x \dot{-} (y+1) = (x \dot{-} y) \dot{-} 1.$$

11.3. Примитивно рекурсивные множества

Будем называть множество *примитивно рекурсивным*, если его характеристическая функция примитивно рекурсивна. (Вариант: если оно является множеством нулей примитивно рекурсивной функции; это то же самое, так как можно сделать подстановку в функцию $x \mapsto 1 \dot{-} x$.)

Пересечение и объединение примитивно рекурсивных множеств примитивно рекурсивны (сложим или перемножим функции, множествами нулей которых они являются). Дополнение примитивно рекурсивного множества примитивно рекурсивно. Отождествляя множества со свойствами, можно сказать, что конъюнкции, дизъюнкции и отрицания примитивно рекурсивных свойств будут примитивно рекурсивны.

Свойства $x = y$ и $x \neq y$ примитивно рекурсивны ($x = y$ тогда и только тогда, когда $(x \dot{-} y) + (y \dot{-} x) = 0$).

Функция $f(x)$, заданная соотношением

$$f(x) = [\ \textbf{if}\ R(x)\ \textbf{then}\ g(x)\ \textbf{else}\ h(x)\ \textbf{fi}\ ],$$

будет примитивно рекурсивной, если таковы функции g и h и свойство R. В самом деле, $f(x)$ можно записать как $r(x)g(x)+(1 \dot{-} r(x))h(x)$, где r — характеристическая функция свойства R.

Теперь можно записать формулу для прибавления единицы по модулю n (для чисел, меньших n):

$$x+1 \bmod n = [\ \mathbf{if}\ x+1=n\ \mathbf{then}\ 0\ \mathbf{else}\ x+1\ \mathbf{fi}\ ]$$

После этого функцию $x \bmod n$ (остаток от деления на n) можно определить рекурсивно:

$$\begin{gathered} 0 \bmod n = 0; \\ (x+1) \bmod n = (x \bmod n) + 1 \bmod n. \end{gathered}$$

Покажем, что ограниченные кванторы, применённые к примитивно рекурсивным свойствам (множествам), дают снова примитивно рекурсивные свойства. Это означает, например, что если свойство $R(x,y)$ примитивно рекурсивно, то свойства

$$S(x,z) = (\exists y \leqslant z)\ R(x,y)$$

и

$$T(y,z) = (\forall y \leqslant z)\ R(x,y)$$

также примитивно рекурсивны. Чтобы убедиться в этом, заметим, что для функций ограниченный квантор соответствует перемножению или суммированию: если свойство $R(x,y)$ равносильно $r(x,y)=0$, то

$$S(x,z) \Leftrightarrow \left[\prod_{y=0}^{z} r(x,y) = 0\right].$$

А произведение легко определить рекурсивно:

$$\begin{aligned} &\prod_{y=0}^{0} r(x,y) = r(x,0); \\ &\prod_{y=0}^{t+1} r(x,y) = \left[\prod_{y=0}^{t} r(x,y)\right] \cdot r(x,t+1); \end{aligned}$$

с суммированием можно поступить аналогичным образом.

После этого легко заметить, что свойство «быть простым» примитивно рекурсивно (любое меньшее число либо равно нулю, либо равно 1, либо не является делителем).

Покажем теперь, что если график некоторой функции f примитивно рекурсивен и её значения ограничены сверху некоторой примитивно рекурсивной функцией g, то сама функция f примитивно рекурсивна. В самом деле, если r — характеристическая функция графика, то есть $r(x, y) = 1$ при $y = f(x)$ и $r(x, y) = 0$ при $y \neq f(x)$ (для простоты мы рассматриваем случай функций одного аргумента), то

$$f(x) = \sum_{i=0}^{\infty} y \cdot r(x, y),$$

а суммирование можно ограничить сверху выражением $g(x)$ и воспользоваться примитивной рекурсивностью ограниченной суммы.

Отсюда легко вывести следующее утверждение: если функция g и свойство $R(x, y)$ примитивно рекурсивны, то функция

$$x \mapsto f(x) = \text{наименьшее } y \leqslant g(x), \text{ для которого } R(x, y)$$

(если для некоторого x такого y нет, то полагаем значение функции равным, скажем, $g(x) + 1$) будет примитивно рекурсивной. В самом деле, график функции f легко описать с помощью ограниченных кванторов.

Такой способ определения функции называют *ограниченным оператором минимизации* — в отличие от неограниченного, где нет заранее известной границы $g(x)$. Как мы увидим, в неограниченном случае получающаяся функция не обязана быть примитивно рекурсивной.

Ограниченный оператор минимизации можно использовать, чтобы убедиться, что функция $x \mapsto$ (минимальное простое число, большее x) примитивно рекурсивна (рассуждение Евклида о бесконечности множества простых чисел устанавливает, что это число не превосходит $x! + 1$, а факториал примитивно рекурсивен). После этого функция $n \mapsto$ (n-е простое число) легко определяется с помощью рекурсии.

11.4. Другие виды рекурсии

Слова «рекурсивное определение функции» можно понимать и в более широком смысле, нежели мы это делали (см. выше определение рекурсии, или примитивной рекурсии) — как любой способ задания функции, который связывает значение функции в данной точке с другими её значениями. Как мы увидим ниже при обсуждении функции Аккермана, есть такие схемы рекурсивных определений, которые выводят из класса примитивно рекурсивных функций. Но есть и такие, которые можно свести к рассмотренной нами схеме.

Мы приведём два примера последнего типа: совместное определение нескольких функций и использование произвольных меньших значений аргумента.

Совместная рекурсия. Пусть две одноместные функции f и g заданы соотношениями:

$$\begin{aligned} f(0) &= a, \\ g(0) &= b, \\ f(n+1) &= F(n, f(n), g(n)), \\ g(n+1) &= G(n, f(n), g(n)), \end{aligned}$$

где a и b — некоторые числа, а функции F и G — примитивно рекурсивные функции трёх аргументов. Покажем, что тогда функции f и g примитивно рекурсивны.

Чтобы доказать это, нам потребуется примитивно рекурсивная нумерация пар — такая функция $\langle x, y\rangle \to [x, y]$ (номер пары мы обозначаем квадратными скобками), которая была бы примитивно рекурсивна вместе с двумя обратными функциями (дающими по номеру пары её первый и второй члены). Тогда мы сможем написать рекурсивное определение для функции $h(n) = [f(n), g(n)]$:

$$\begin{aligned} h(0) =& [a, b], \\ h(n+1) =& [F(n, p_1(h(n)), p_2(h(n))), \\ & G(n, p_1(h(n)), p_2(h(n)))], \end{aligned}$$

где функции p_1 и p_2 дают по номеру пары первый и второй её члены. Если функция h примитивно рекурсивна, то и функции f и g (композиции h с функциями p_1 и p_2) также примитивно рекурсивны.

Осталось объяснить, как найти примитивно рекурсивную нумерацию пар. Можно заметить, что есть многочлен второго порядка с двумя переменными, задающий взаимно однозначное соответствие $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Это соответствие видно из таблицы:

$$\begin{array}{llll} 6 & & & \\ 3 & 7 & & \\ 1 & 4 & 8 & \\ 0 & 2 & 5 & 9 \end{array}$$

Примитивную рекурсивность обратных отображений p_1 и p_2 можно установить, воспользовавшись ограниченной минимизацией, так как $p_1(n)$ есть минимальное $x \leqslant n$, для которого найдётся $y \leqslant n$, при котором $[x, y] = n$.

Менее симметричная нумерация пар может быть задана формулой $[a, b] = (2a+1)2^b$. Можно также заметить, что нам не нужно, чтобы все числа были номерами каких-то пар, и воспользоваться нумерацией $[a, b] = 2^a 3^b$.

Заметим в заключение, что аналогичная конструкция применима для большего числа одновременно определяемых функций и для функций от большего числа аргументов.

Возвратная рекурсия. Следующее утверждение показывает, что при рекурсивном определении можно использовать не только значение в предыдущей точке, но и любое предшествующее значение.

Теорема 74. Пусть функция g одного аргумента примитивно рекурсивна, причём $g(x) < x$ при $x > 0$; пусть

F — примитивно рекурсивная функция двух аргументов; пусть c — произвольная константа. Тогда функция h, определённая соотношениями

$$h(0) = c,$$
$$h(x) = F(x, h(g(x))) \text{ при } x > 0$$

примитивно рекурсивна.

$\lhd$ Чтобы доказать эту теорему, используем следующую нумерацию конечных последовательностей натуральных чисел: номером пустой последовательности считаем число 1, номером одноэлементной последовательности $\langle a \rangle$ считаем число 2^{a+1}, последовательность $\langle a, b \rangle$ имеет номер $2^{a+1}3^{b+1}$, последовательность $\langle a, b, c \rangle$ имеет номер $2^{a+1}3^{b+1}5^{c+1}$ и так далее (основания степеней — простые числа). Будем обозначать номер последовательности $\langle a, b, \ldots, z \rangle$ через $[a, b, \ldots, z]$. Эта нумерация в некотором смысле примитивно рекурсивна. Конечно, буквально это понимать нельзя, так как нумерация представляет собой «функцию с переменным числом аргументов». Но разные связанные с ней функции примитивно рекурсивны. В частности, таковы функции

- $\mathrm{Length}(x)$ = длина последовательности с номером x;
- $\mathrm{Select}(i, x)$ = i-ый член последовательности с номером x;
- $\mathrm{Append}(x, y)$ = номер последовательности, которая получается приписыванием числа y к последовательности с номером x.

Все эти функции (и другие аналогичные) сводятся к различным операциям с простыми числами и множителями, которые мы в сущности уже разбирали.

Теперь мы докажем, что функция

$$x \mapsto H(x) = [h(0), h(1), \ldots, h(x)]$$

примитивно рекурсивна. В самом деле, $H(0) = [c]$, а

$$H(k+1) = \mathrm{Append}(H(k), F(k+1, \mathrm{Select}(g(k+1), H(k)))). \rhd$$

11.5. Машины Тьюринга и примитивно рекурсивные функции

Мы рассмотрели несколько различных приёмов построения примитивно рекурсивных функций. Тем не менее остаётся не вполне ясным, насколько этот класс широк. Сейчас мы покажем, что он включает в себя все достаточно быстро вычислимые функции.

Теорема 75. Любая функция, вычислимая на машине Тьюринга не более чем за примитивно рекурсивное (от длины входа) время, примитивно рекурсивна.

$\lhd$ Напомним, что мы считаем входом и выходом машины Тьюринга слова из нулей и единиц. Поскольку аргументами и значениями примитивно рекурсивных функций являются числа, теорема будет иметь смысл, только если мы договоримся отождествлять числа и слова. Как уже говорилось, мы отождествляем число n со словом, которое получается после удаления старшего бита 1 в двоичном разложении числа $n+1$.

При имитации работы машин Тьюринга с помощью программ мы кодировали состояние машины четырьмя числами (код левой части ленты, код правой части ленты, состояние и буква под головкой). При этом удобно было такое кодирование: левую часть ленты мы считали записью числа в системе счисления, в которой основание равно числу символов в алфавите машины, а пробел считается нулём; с правой частью ленты мы поступали так же, только в обратном порядке (младшие разряды у головки). При этом добавление или изъятие символа у головки соответствовало простой арифметической операции (удаление — это деление нацело, добавление — умножение на основание системы счисления и сложение). При таком кодировании функции перехода (четыре функции четырёх аргументов, показывающие следующее состояние как функцию предыдущего), записываются простыми формулами и примитивно рекурсивны.

Теперь рассмотрим итерированную функцию перехода, которая говорит, каково будет состояние машины Тьюринга после t шагов. Точнее, тут имеются четыре функции от пяти аргументов (первые четыре аргумента кодируют состояние, пятый представляет собой число шагов). Их определение имеет вид совместной рекурсии, которую мы только что разобрали. Поэтому эти функции примитивно рекурсивны. Будем считать, что после появления заключительного состоянии конфигурация машины не меняется. Если мы знаем, что число шагов работы ограничено примитивно рекурсивной функцией, то достаточно подставить её на место пятого аргумента (числа шагов), чтобы убедиться, что заключительная конфигурация машины является примитивно рекурсивной функцией от её начальной конфигурации. Следовательно, результат работы является примитивно рекурсивной функций начального данного.

Это рассуждение неявно использует примитивную рекурсивность различных функций, связанных с переходом от одного представления данных к другому. Например, вход машины Тьюринга является двоичным словом, которое мы договорились отождествлять с некоторым числом x. Этому входу соответствует начальная конфигурация машины Тьюринга, которую мы кодируем четвёркой чисел. Нам важно, что эта четвёрка примитивно рекурсивно зависит от x. Это легко понять, так как преобразование связано с переходом от одной системы счисления к другой (одно и то же слово кодирует разные числа в разных системах счисления); примитивную рекурсивность таких функций легко установить с помощью описанных выше методов. Кроме того, нам надо из выходной конфигурации примитивно рекурсивно извлечь результат и также перекодировать его, а также по входу получить его длину (чтобы подставить в примитивно рекурсивную функцию, ограничивающую число шагов). Но всё это также не выходит из круга разобранных выше приёмов, и подробно останавливаться на этом мы не будем. $\triangleright$

Эта теорема убеждает нас в примитивной рекурсивности многих довольно сложно определяемых функций. Например, рассмотрим функцию $n \mapsto$ (n-ый десятичный знак числа π). Известно, что вычислены миллионы таких знаков, поэтому есть все основания полагать, что известные алгоритмы работают не слишком долго — было бы очень странно, если бы время их работы (даже учитывая неудобство машины Тьюринга для программирования) не оценивалось бы, скажем, функцией $c \times 2^n$ при достаточно большом c. А такая оценка примитивно рекурсивна, что позволяет сослаться на только что доказанную теорему. (На самом деле тут большой запас — существуют примитивно рекурсивные функции, которые растут гораздо быстрее 2^n.)

11.6. Частично рекурсивные функции

Операторы примитивной рекурсии и подстановки не выводят нас из класса всюду определённых функций. Не так обстоит дело с *оператором минимизации*, о котором мы уже упоминали. Он применяется к $(k+1)$-местной функции f и даёт k-местную функцию g, определяемую так: $g(x_1, \ldots, x_k)$ есть *наименьшее* y, *для которого* $f(x_1, \ldots, x_k, y) = 0$.

Смысл выделенных слов ясен, если функция f всюду определена. Если нет, то понимать их надо так: значение $g(x_1, \ldots, x_k)$ равно y, если $f(x_1, \ldots, x_k, y)$ определено и равно нулю, а все значения $f(x_1, \ldots, y_k, y')$ при $y' < y$ определены и не равны нулю.

Часто используется обозначение

$$g(x_1, \ldots, x_k) = \mu y\ (f(x_1, \ldots, x_k, y) = 0),$$

и потому оператор минимизации также называют *μ-оператором*.

Ясно, что такое определение обеспечивает вычислимость g, если вычислима f (мы перебираем в порядке возрастания все y, ожидая появления нулевого значения).

86. Покажите, что если изменить определение и разрешить $f(x_1, \ldots, x_k, y')$ быть не определённым при $y' < y$, то функция g может быть невычислимой при вычислимой f.

Функции, получающиеся из базисных (нуля, проекции и прибавления единицы) с помощью операторов подстановки, примитивной рекурсии и минимизации, называются *частично рекурсивными*. Если такая функция оказывается всюду определённой, то её называют *общерекурсивной* функцией.

Такая странная терминология, видимо, сложилась в результате переводов с английского. По-английски были "primitive recursive functions" (примитивно рекурсивные функции). Затем было дано более общее (general) понятие вычислимой всюду определённой функции, и такие функции были названы "general recursive functions". Затем стали рассматривать и частичные (partial) вычислимые функции, называя их "partial recursive functions". В русском же языке слово "partial" попало не туда, и вместо частичных рекурсивных функций стали говорить о частично рекурсивных функциях. Та же участь постигла и слово "general", которое странным образом вошло в прилагательное «общерекурсивная» и означает, что функция всюду определена.

Теорема 76. Всякая функция, вычислимая с помощью машины Тьюринга, является частично рекурсивной.

$\lhd$ Пусть f — вычислимая с помощью машины Тьюринга (обозначим эту машину через M) функция одного аргумента. Рассмотрим свойство $T(x, y, t)$, состоящее в том, что машина M на входе x даёт ответ y за время не более чем t. Как мы видели выше, по входу машины Тьюринга и по времени t можно примитивно рекурсивно вычислить её состояние в момент t; ясно, что можно также узнать, закончила ли она работу, и если да, то был ли ответ равен y. Итак, свойство T примитивно рекурсивно.

Теперь объединим аргументы y и t в пару с помощью примитивно рекурсивной нумерации; получится примитивно рекурсивная функция T', для которой $T'(x, [y, t]) = T(x, y, t)$; теперь можно написать $f(x) = p_1(\mu z T'(x, z))$, где p_1 даёт по номеру пары её первый член, а μz означает «наименьшее z, для которого...». Таким образом, функция f является частично рекурсивной. $\rhd$

Верно и обратное:

Теорема 77. Всякая частично рекурсивная функция вычислима на машине Тьюринга.

$\lhd$ Легко написать программу с конечным числом переменных, вычисляющую любую частично рекурсивную функцию (подстановка сводится к последовательному выполнению программ, рекурсия — к циклу типа **for**, минимизация — к циклу типа **while**; оба вида циклов легко реализуются с помощью операторов перехода).

После этого остаётся только сослаться на то, что всякая функция, вычисляемая программой с конечным числом регистров, вычислима на машине Тьюринга (как мы видели в разделе 10.2, теорема 66). $\triangleright$

Поэтому если мы верим в «тезис Тьюринга», гласящий, что всякая вычислимая функция вычислима на машине Тьюринга, то должны верить и в «тезис Чёрча» (всякая вычислимая функция частично рекурсивна), так что эти тезисы равносильны.

На самом деле история здесь сложнее и примерно может быть описана так. Определение примитивно рекурсивной функции было дано великим логиком Куртом Гёделем и использовано как техническое средство при доказательстве теоремы Гёделя о неполноте, в начале 1930-х годов. Он же дал определение общерекурсивной функции (не совпадающее с приведённым нами, но эквивалентное ему). Американский логик Алонзо Чёрч сформулировал свой тезис для всюду определённых функций, предположив, что всякая вычислимая всюду определённая функция является общерекурсивной. Затем американский математик Клини предложил распространить этот тезис на функции, не являющиеся всюду определёнными.

Параллельно английский математик Тьюринг и американский математик Пост предложили свои модели абстрактных вычислительных машин (машины Тьюринга и Поста), отличающиеся лишь некоторыми деталями, и высказали предположение о том, что такие машины покрывают весь класс алгоритмических процессов. Вскоре стало ясно, что вычислимость функции на таких машинах равносильна частичной рекурсивности. (Исторические подробности можно узнать из книги Клини [4].)

Так или иначе, сейчас выражения «тезис Тьюринга», «тезис Чёрча», «тезис Поста» и т. п. обычно употребляют как синонимы: эти тезисы утверждают, что интуитивно понимаемый класс вычислимых частичных функций совпадает с формально определённым классом частично рекурсивных функций (тезис Чёрча), вычислимых на машинах Тьюринга функций (тезис Тьюринга) и т. д. При таком понимании можно доказать равносильность всех этих тезисов, так как все известные формальные определения вычислимости (частичная рекурсивность, машины Тьюринга и т. д.) приводят к одному и тому же классу функций.

(Заметим в скобках, что ещё одна вычислительная модель — *нормальные алгорифмы*, или *алгоритмы Маркова*, были предложены Андреем Андреевичем Марковым-младшим (сыном старшего, в честь которого названы марковские цепи и марковские процессы). Это было уже в 1950-ые годы. Марков писал слово *алгорифм* через «*ф*». Соответствующий принцип (всякий алгорифм эквивалентен нормальному) он называл *принципом нормализации*. В его работах нормальные алгорифмы использовались для построения неразрешимого ассоциативного исчисления (см. раздел 9.4). Кроме того, Марков реально выписал во всех деталях конструкцию универсального алгоритма и провёл точное доказательство его корректности; кажется, это достижение остаётся никем не повторённым — ни у кого больше не хватило настойчивости, чтобы для какого-то языка программирования написать на этом языке его интерпретатор и формально доказать его корректность.)

Наше доказательство теорем 76 и 77 позволяет также получить такое следствие, называемое иногда *теоремой Клини о нормальной форме*:

Теорема 78. Всякая частично рекурсивная функция f представима в виде

$$f(x) = a(\mu z(b(x,z) = 0)),$$

где a и b — некоторые примитивно рекурсивные функции.

◁ В самом деле, любая частично рекурсивная функция вычислима на машине Тьюринга, а следовательно, представима в нужном нам виде, как видно из доказательства теоремы 76 (в качестве a берётся функция, дающий первый член пары по её номеру). ▷

Мы сформулировали эту теорему для случая одноместной функции f, но аналогичное утверждение верно и для функций нескольких аргументов (и доказательство почти не меняется).

87. Покажите, что одним μ-оператором, применяя его последним, не обойтись: не всякая частично рекурсивная функция представима в виде

$$f(x) = \mu z(b(x, z) = 0)$$

где b — некоторая примитивно рекурсивная функция.

Из теоремы Клини о нормальной форме вытекает такое утверждение:

Теорема 79. Всякое перечислимое множество есть проекция примитивно рекурсивного множества.

$\triangleleft$ Перечислимое множество есть область определения рекурсивной функции; представив её в нормальной форме, видим, что область определения есть проекция множества $\{\langle x, z\rangle \mid b(x, z) = 0\}$. $\triangleright$

11.7. Вычислимость с оракулом

Определение класса частично рекурсивных функций легко модифицировать для случая вычислимости с оракулом. Пусть имеется некоторая всюду определённая функция α. Рассмотрим класс $\mathcal{F}[\alpha]$, который состоит из базисных функций, функции α и всех других функций, которые могут быть из них получены с помощью подстановки, примитивной рекурсии и минимизации.

(Формально можно сказать так: $\mathcal{F}[\alpha]$ есть минимальный класс, который содержит базисные функции, функцию α и замкнут относительно подстановки, примитивной рекурсии и минимизации. Такой минимальный класс существует — достаточно взять пересечение всех классов с этими свойствами.)

Теорема 80. Класс $\mathcal{F}[\alpha]$ состоит из всех функций, вычислимых с оракулом α (то есть с помощью программ, вызывающих α как внешнюю процедуру).

$\triangleleft$ Прежде всего заметим, что все функции из класса $\mathcal{F}[\alpha]$ вычислимы с помощью таких программ. Это можно объяснить, например, так. Программы с конечным числом переменных вычисляли все частично-рекурсивные функции. Если добавить к ним примитивную операцию вычисления значения функции α в заданной точке, то они точно так же будут вычислять все функции из класса $\mathcal{F}[\alpha]$.

Более содержательно обратное утверждение: мы хотим доказать, что если некоторая (частичная, вообще говоря) функция вычислима относительно α, то она может быть получена из базисных функций и из α с помощью подстановки, рекурсии и минимизации.

Для этого вспомним критерий относительной вычислимости, данный нами выше в разделе 7.2 (теорема 45). Пусть функция f вычислима относительно всюду определённой функции α. Тогда, как мы знаем, существует перечислимое множество W троек вида $\langle x, y, t\rangle$, где x и y — натуральные числа, а t — образцы (функции с конечной областью определения), которое является корректным (тройки с одинаковым x, но разными y содержат несовместные образцы) и для которого

$$f(x) = y \Leftrightarrow \exists t\ (\langle x, y, t\rangle \in W \text{ и } t \text{ есть часть функции } \alpha).$$

Мы сейчас покажем, что свойство «t есть часть функции α» является примитивно рекурсивным относительно α: его характеристическая функция получается из базисных функций и из α с помощью операций подстановки и рекурсии. (Напомним, что мы отождествляем образцы с их номерами в какой-то нумерации; о её выборе см. ниже.)

После этого останется записать W как проекцию примитивно рекурсивного множества ($\langle x, y, t\rangle \in W \Leftrightarrow$ $\Leftrightarrow \exists u\,(v(x, y, t, u) = 0)$, где v примитивно рекурсивна), и заметить, что

$$f(x) = p_1(\mu z\ v'(x, z) = 0).$$

Здесь v' — примитивно рекурсивная относительно α функция, для которой $v'(x, [y, t, u]) = 0$ тогда и только тогда, когда $v(x, y, t, u) = 0$ и t есть часть функции α, а функция p_1 выделяет из номера тройки $[y, t, u]$ её первый член y.

Осталось показать, что множество $\{t \mid \text{образец с номером } t \text{ есть часть функции } \alpha\}$ является примитивно рекурсивным относительно α. При доказательстве мы будем предполагать, что нумерация образцов такова, что следующие функции примитивно рекурсивны (они определены для образцов с непустой областью определения; для пустого образца доопределим их как угодно):

- last-x(t) — наибольшее из чисел, на котором определён образец с номером t;
- last-y(t) — значение функции-образца с номером t в максимальной точке своей области определения;
- all-but-last(t) — номер образца, который получится из образца с номером t, если удалить максимальную точку области определения.

Тогда можно записать такое рекурсивное определение: образец с номером t есть часть функции α, если либо этот образец пуст, либо $\alpha(\text{last-x}(t)) = \text{last-y}(t)$ и образец с номером all-but-last(t) есть часть функции α.

Это определение использует «возвратную рекурсию», которая разобрана в теореме 74 (с. 151); значение функции определяется рекурсивно через её значения в меньших точках. Надо только выбрать нумерацию образцов так, чтобы all-but-last(t) было меньше t для всех t. Этого несложно добиться: например, можно нумеровать образцы с помощью простых чисел, считая, что образец $\{\langle a,b\rangle,\ldots,\langle e,f\rangle\}$ имеет номер $p_a^{(b+1)}\ldots p_e^{(f+1)}$, где p_i — простое число с номером i (так что $p_0 = 2$, $p_1 = 3$, $p_2 = 5,\ldots$). $\triangleright$

Заметим, что доказательство стало бы немного проще, если использовать только «сплошные образцы» (областью определения которых является некоторый начальный отрезок натурального ряда) — тогда можно начать с того, что установить примитивную рекурсивность (относительно α) функции $n \mapsto [\alpha(0),\alpha(1),\ldots,\alpha(n)]$. Легко понять также, что в определении относительной вычислимости можно ограничиться только сплошными образцами.

11.8. Оценки скорости роста. Функция Аккермана

Обратимся теперь к вопросу, который можно было бы задать уже давно: существуют ли общерекурсивные, но не примитивно рекурсивные функции? Мы приведём два доказательства существования таковых. Первое исходит из общих соображений:

Теорема 81. Существует всюду определённая вычислимая функция двух аргументов, универсальная для класса всех примитивно рекурсивных функций одного аргумента.

Очевидно, что если U — такая функция, то функция d, для которой $d(n) = U(n, n) + 1$, будет всюду определённой, вычислимой и будет отличаться от любой примитивно рекурсивной функции (от n-ой — в точке n).

$\triangleleft$ Всякая примитивно рекурсивная функция получается из базисных с помощью некоторой последовательности операций подстановки и рекурсии. Ясно, что такую последовательность можно описать словом в конечном алфавите — так сказать, программой (в которой последовательно определяются различные примитивно рекурсивные функции и для каждой написано, из каких других она получается и с помощью каких операций). Из всех программ отберём программы для одноместных функций (разумеется, в качестве промежуточных функций можно использовать функции с любым числом аргументов). Множество таких программ разрешимо, их можно пронумеровать вычислимым образом. Функция $\langle n, x\rangle \mapsto$ $\mapsto$ (результат применения функции, заданной программой номер n, к числу x) будет вычислима и по построению будет универсальной для класса примитивно рекурсивных функций. $\triangleright$

Однако интересно указать и более конкретную причину, мешающую некоторым вычислимым функциям быть примитивно рекурсивными. Вот одна из возможностей: примитивно рекурсивные функции не могут быстро расти. Эта идея восходит к Аккерману, который построил функцию, растущую быстрее всех примитивно рекурсивных — *функцию Аккермана*. Сейчас мы изложим эту конструкцию (хотя детали построения будут иными).

Определим последовательность функций $\alpha_0, \alpha_1, \ldots$ от одного аргумента. (Все эти функции будут всюду определёнными.) Положим $\alpha_0(x) = x + 1$. Определяя α_i, мы будем использовать такое обозначение: $f^{[n]}(x)$ означает $f(f(\ldots f(x)\ldots))$, где функция f использована n раз. Так вот,

$$\alpha_i(x) = \alpha_{i-1}^{[x+2]}(x)$$

(почему удобно применять функцию α_{i-1} ровно $x+2$ раза, мы увидим чуть позже).

Очевидные свойства (формально их можно доказать по индукции):

- $\alpha_i(x) > x$ при всех i и x;
- $\alpha_i(x)$ возрастает с возрастанием x;
- $\alpha_i(x)$ возрастает с возрастанием i (для каждого фиксированного x);
- $\alpha_i(x) \geqslant \alpha_{i-1}(\alpha_{i-1}(x))$.

Теперь можно оценить скорость роста любой примитивно рекурсивной функции.

Теорема 82. Пусть f — примитивно рекурсивная функция n аргументов. Тогда найдётся такое k, что

$$f(x_1, \ldots, x_n) \leqslant \alpha_k(\max(x_1, \ldots, x_n))$$

при всех $x_1, \ldots, x_n$.

$\triangleleft$ Идея проста — можно оценить скорость роста композиции функций, зная оценки для каждой из них; аналогично для рекурсии. Формально говоря, доказательство использует «индукцию по построению» примитивно рекурсивных функций.

Для базисных функций утверждение очевидно. Посмотрим на подстановку. Пусть

$$f(x) = g(h_1(x), \ldots, h_k(x))$$

(для краткости мы пишем одну букву x, имея в виду вектор переменных). Пусть α_N оценивает все функции $h_1, \ldots, h_k$ и функцию g сверху, то есть $h_i(x) \leqslant \alpha_N(\max(x))$ при всех i и x, а также $g(y) \leqslant \alpha_N(\max(y))$ (здесь $\max(u)$ означает максимальный элемент в наборе u). Тогда $f(x)$ не превосходит

$$\alpha_N(\max(h_1(x), \ldots, h_k(x))) \leqslant \alpha_N(\alpha_N(x)) \leqslant \alpha_{N+1}(x)$$

(мы пользуемся указанными выше свойствами функций α_i).

Похоже (но немного сложнее) дело обстоит с рекурсией. Пусть функция f определяется рекурсивно:

$$\begin{aligned} f(x, 0) &= g(x); \\ f(x, n+1) &= h(x, n, f(x, n)). \end{aligned}$$

(Здесь x также обозначает набор нескольких переменных.) Пусть функции g и h оцениваются сверху функцией α_N. Тогда

$$\begin{aligned} f(x, 1) = h(x, 0, f(x, 0)) &\leqslant \alpha_N(\max(x, 0, f(x, 0))) \leqslant \\ &\leqslant \alpha_N(\max(x, 0, \alpha_N(\max(x)))) \leqslant \alpha_N(\alpha_N(\max(x))) \end{aligned}$$

(в последнем переходе мы пользуемся тем, что $\alpha_N(t) > t$). Аналогично $f(x, 2) \leqslant \alpha_N(\alpha_N(\alpha_N(\max(x))))$ и вообще

$$f(x, i) \leqslant \alpha_N^{[i+1]}(\max(x)) \leqslant \alpha_{N+1}(\max(i, \max(x))),$$

что и требовалось доказать. $\triangleright$

Заметим, что каждый оператор подстановки или рекурсии увеличивает номер верхней оценки на 1, так что функция, в определении которой не более 100 операторов, растёт не быстрее α_{101}.

Очевидным следствием полученной оценки является такое утверждение:

Теорема 83. Функция $A(n) = \alpha_n(n)$ растёт быстрее любой примитивно рекурсивной функции.

Отметим, что определение функции Аккермана (точнее, функции $\langle n, x \rangle \mapsto \alpha_n(x)$) вполне можно назвать рекурсивным — одно значение этой функции определяется через другие, с меньшим первым аргументом. Оно является примером рекурсивного определения, не сводящегося к примитивной рекурсии.

88. Покажите, что прямой пересчёт (в возрастающем порядке) бесконечного примитивно рекурсивного множества может не быть примитивно рекурсивным.

89. Покажите, что функция, обратная к примитивно рекурсивной биекции $i\colon \mathbb{N} \to \mathbb{N}$, может не быть примитивно рекурсивной.

Литература

[1] Дж. Булос, Р. Джеффри, *Вычислимость и логика.* Перевод с английского В. А. Душского и Е. Ю. Ногиной под редакцией С. Н. Артёмова. М.: Мир, 1994. 396 с.

[2] Ю. Л. Ершов, *Теория нумераций.* М.: Наука, 1977. 416 с. (Серия: Математическая логика и основания математики.)

[3] Н. Катленд, *Вычислимость. Введение в теорию рекурсивных функций.* Перевод с английского А. А. Мучника под редакцией С. Ю. Маслова. М.: Мир, 1983. 256 с.

[4] С. К. Клини, *Введение в метаматематику.* Перевод с английского А. С. Есенина-Вольпина под редакцией В. А. Успенского. М.: Издательство иностранной литературы, 1957. 526 с.

[5] А. И. Мальцев, *Алгоритмы и рекурсивные функции.* М.: Наука, 1965. 392 с.

[6] Ю. И. Манин, *Вычислимое и невычислимое.* М.: Советское радио, 1980. 128 с.

[7] М. Минский, *Вычисления и автоматы.* Перевод с английского Б. Л. Овсиевича и Л. Я. Розенблюма. М.: Мир, 1971. 366 с.

[8] Х. Роджерс, *Теория рекурсивных функций и эффективная вычислимость.* Перевод с английского В. А. Душского, М. И. Кановича и Е. Ю. Ногиной. Под редакцией В. А. Успенского. М.: Мир, 1972. 624 с.

[9] *Справочная книга по математической логике в четырёх частях под редакцией Дж. Барвайса. Часть III. Теория рекурсии.* Перевод с английского С. Г. Дворникова, И. А. Лаврова. Под редакцией Ю. Л. Ершова. М.: Наука, 1982. 360 с.

[10] В. А. Успенский, *Лекции о вычислимых функциях.* М.: Физматгиз, 1960. 492 с. (Серия: Математическая логика и основания математики.)

[11] В. А. Успенский, А. Л. Семёнов, *Теория алгоритмов: основные открытия и приложения.* М.: Наука, 1987. (Серия: Библиотечка программиста.)

[12] Дж. Шенфилд, *Степени неразрешимости.* Перевод с английского И. А. Лаврова. Под редакцией Ю. Л. Ершова. М.: Наука, 1977. 192 с. (Серия: Математическая логика и основания математики.) Книга включает в себя в качестве приложения переводы статей К. Е. М. Ейтса, А, Х. Лахлана и Л. Фейнера.

Предметный указатель

α-вычислимая функция 81, 159
α-перечислимое множество 82
β-функция Гёделя 136, 144
$\mathbf{0}'$-вычислимая функция 85
$\mathbf{0}, \mathbf{0}', \ldots, \mathbf{0}^{(n)}$ 104
$\mathcal{F}[\alpha]$ 159
$\mathrm{Disjoint}(A)$ 106
$\exists x$, $\forall x$ 98
$\Gamma(T)$ 42
$\Gamma(t)$ 41
$\wedge$, $\vee$, $\neg$ 98
$\leqslant_m$ 59
$\leqslant_T$ 77
μ-оператор 155
$\mathrm{Subset}(A)$ 105
π 10
Π_2 109
Π_n 98–100, 138, 143
$\mathrm{Proof}(x, y)$ 142
Σ_n 98–100, 104, 138, 143
A-вычислимая функция 103
A-перечислимое множество 103
A-разрешимое множество 103
D_x 105
e 10
m-полное множество 61, 65, 66, 74, 75, 103
m-сводимость 59, 74, 100, 139
  пар 73
m-сводящая функция 59
m-степень 103
m-эквивалентные множества 103
T-сводимость 77
T-степень 103
T-эквивалентные множества 103

ExecuteProgram, процедура 51

GetProgramText, процедура 51

Адекватность 140
Аккермана функция 163
Алгоритм (алгорифм) Маркова 158
Алфавит 113, 118, 126
  сокращение 116
Арифметическая иерархия 98, 138, 142, 143
Арифметическая формула 134
Арифметическая функция 134
Арифметическое множество 129, 134, 135, 138, 139, 143
Ассоциативное исчисление 118
  двустороннее 124, 127
  неразрешимое 119, 123, 158

Базисная функция 146
Бейсик 50
Буква 118

Взаимно простые числа 136

Возвратная рекурсия 151
Время вычисления 153
Вход машины Тьюринга 114
Выигрышное условие 94
Вычислимая нумерация 24, 30
Вычислимая перестановка 66
Вычислимая функция 8, 11, 129, 157
Вычислимо изоморфные множества 56
Вычислимое действительное число 16, 21
Вычислимость
  на машине с конечным числом регистров 131
  на машине Тьюринга 115, 132
  относительно α 81
  с оракулом 76, 159
Вычитание усечённое, рекурсивное определение 147

Гёделя теорема 140, 141
Главная (гёделева) нумерация 24, 26, 32
Главная универсальная функция 25, 28
  относительно α 84
Главное универсальное множество 29, 30, 61
Головка 113
Гомоморфизм 126
График функции 14

Двустороннее ассоциативное исчисление 124, 127
Декартово произведение множеств 13
Десятая проблема Гильберта 143
Диагональная функция 47
Диагональное сечение 21, 61, 73
Дизъюнкция 98, 100
Диофантово уравнение 15
Доказательство 141
Дополнение 12, 99

Заключительное состояние 114

Изоморфизм
  m-полных множеств 66
  m-полных пар 75
  главных нумераций 39
  универсальных множеств 56
Иммунное множество 22, 23, 62, 65
Исчисление 140
  ассоциативное 118
  неразрешимое 119, 123, 158

Кванторы 98, 142
  ограниченные 148
Китайская теорема об остатках 137
Композиция функций 24, 27
Конечно порождённая полугруппа 126
Конечное множество
  номер 105
Конкатенация 126
Конструктивный объект 8
Конфигурация машины Тьюринга 114, 120, 132
Конъюнкция 98, 100
Коперечислимость 99

Корректное множество 79, 160
Креативное множество 71

Лента 113

Массивы, моделирование 131
Машина
  Поста 157
  с конечным числом регистров 129
  с оракулом 76
  Тьюринга 6, 112, 113, 115, 117, 125, 132, 153, 156, 157
Минимизации оператор 149, 155
Множества
  m-эквивалентные 103
  T-эквивалентные 103
  неотделимость 21, 72
  эффективная неотделимость 73, 74
Множество
  A-перечислимое 103
  A-разрешимое 103
  m-полное 61, 65, 103
  арифметическое 129, 134, 135, 138, 139, 143
  иммунное 22, 23, 62, 65
  креативное 71
  неразрешимое 10
  перечислимое 10, 12, 13, 15, 98, 138, 143
  перечислимое неразрешимое 20, 104, 123
  примитивно рекурсивное 147
  продуктивное 69
  простое 22, 66
  разрешимое 9, 13, 15, 98, 138
  универсальное 18
  универсальное в Σ_n/Π_n 101
  эффективно неперечислимое 63
Множество номеров 36, 109
  нигде не определённой функции 32
Моделирование 120

Натуральная нумерация 24
Начальное состояние 113
Недетерминированный алгоритм 13
Неотделимые множества 22, 72
Неподвижная точка 45, 142
Неразрешимое ассоциативное исчисление 119, 123, 158
Неразрешимое множество 10
Несравнимые по Тьюрингу множества 89
Нигде не определённая функция 109
Нижняя точка 21
Номер 24
  пары 26
  функции 109
Нормализации принцип 158
Нормальный алгорифм 158
Нумерация 24
  вычислимая 24, 30
  главная (гёделева) 24, 26, 32
  конечных множеств 105

конечных последовательностей 152
натуральная 24
однозначная 35

Область значений 11
Область определения 11
Образ множества 15
Образец 41, 79, 107, 160
Образующие полугруппы 125, 127
Общерекурсивная функция 156, 157
Объединение множеств 12
Ограниченный квантор 148
Ограниченный оператор минимизации 149
Однозначная нумерация 35
Оператор минимизации 149, 155
Операция скачка 102, 103
Оракул 76
Отделяющее множество 22
Относительная вычислимость 77, 159
Отрицание 98, 99

Пар нумерация 26
Парадокс лжеца 142
Паскаль 40, 112, 117
Пересечение множеств 12
Пересечение перечислимых множеств 31
Перечислимое множество 10, 12, 13, 15, 98, 138, 143
универсальное 101
Перечислимое неразрешимое множество 20, 104, 123
Перечислимое снизу/сверху число 16, 21
Перечислимость 14
Перечислимость относительно α 82
Перечислимые неотделимые множества 21, 37
Перечислимые свойства функций 41
Подслово 119
Подстановка 145
Полнота 140
Полугруппа 125, 127
Полухарактеристическая функция 11, 59
Поста машина 157
Поста проблема 91
Поста тезис 157
Поста теорема 108
Правило 118
Представление множества формулой 135
Преобразование слов 119
Примитивно рекурсивная функция 145, 146, 153, 157
Примитивно рекурсивное множество 147
Принцип нормализации 158
Пробела символ 113
Проблема
остановки 112
Поста 91
равенства слов 112
самоприменимости 20
Программа с конечным числом переменных 132
Программа, печатающая свой текст 47
Продолжение образца 41
Продуктивное множество 69
Проекция 14, 99

Прообраз 15, 60
Простое множество 22, 66
Простое число 131, 149, 150
Противоречивые тройки 79
Псевдообратная функция 16
Пустое слово 126
Пустой символ 113

Разрешимое множество 9, 13, 15, 98, 138
Разрешимость 14
Результат работы 114
Рекурсивная функция 145
Рекурсия 145, 150
  возвратная 151
  примитивная 145
  совместная 150
Релятивизация 81
Роджерса теорема 39, 58

Сведе́ние 32
Свободная полугруппа 126
Сводимость нумераций 32
Сводимость по Тьюрингу 77, 88
Сечение
  диагональное 21
  множества 18
  функции 17
Си 40
Сильно главная нумерация 84
Символ 113, 118
Скачок 103
Слово 118, 126
Сложение, рекурсивное определение 146
Совместная рекурсия 150
Совместные образцы 79
Соотношения в полугруппе 125–127
Состояние 113, 118
  заключительное 114
  начальное 113
Стек 132
Степень
  неразрешимости 103
Счётчик команд 135

Таблица переходов 113, 114, 121
Тарского теорема 139, 140
Тезис Тьюринга 117
Теорема
  Гёделя о неполноте 140, 141
  Клини о неподвижной точке (о рекурсии) 45
  Клини о нормальной форме 158
  Мучника – Фридберга 92
  о неподвижной точке для множеств 55
  о неподвижной точке с параметром 54
  об униформизации 15
  Поста 13, 33, 108
  Роджерса 39, 58
  Тарского 139, 140
  Успенского – Райса 33, 109
Транслятор 40
Тьюринга машина 6, 112, 113, 115, 117, 125, 132, 138, 153, 156, 157
Тьюринга тезис 117, 138, 157
Тьюрингова степень 103

Удвоение слова 115
Указание в методе приоритета 92
Умножение, рекурсивное определение 146

Универсальная вычислимая функция 17, 19
Универсальная функция 17, 158, 162
неглавная 35
Универсальная функция трёх аргументов 25
Универсальное в Σ_n/Π_n множество 101
Универсальное множество 18
Универсальное перечислимое множество 18, 101
Успенского – Райса теорема 33, 109

Формула
арифметики 129, 134
представляющая множество 135
Фрагмент 89
Функция
α-вычислимая 81
$\mathbf{0}'$-вычислимая 85
A-вычислимая 103
m-сводящая 59
Аккермана 150, 163
арифметическая 134
вычислимая 8, 11, 129, 157
вычислимая на машине Тьюринга 115, 156
диагональная 47
нигде не определённая 109
общерекурсивная 156, 157
полухарактеристическая 11, 59
примитивно рекурсивная 145, 146, 153, 157
рекурсивная 145
универсальная 17, 158, 162
универсальная вычислимая 17, 19
характеристическая 9, 59
частично рекурсивная 155, 156

Характеристическая функция 9, 59

Частично рекурсивная функция 155, 156
Чёрча тезис 157
Число
вычислимое 16, 21
перечислимое снизу/сверху 16, 21

Эквивалентность по Тьюрингу 103
Элемент 92
Эффективно неотделимые множества 73, 74
Эффективно неперечислимое множество 63

Язык программирования 25

Указатель имён

Аккерман В. (W. Ackermann) 150, 163
Гёдель К. (K. Gödel, 1906 – 1978) 136, 144, 157
Гильберт Д. (D. Hilbert, 1862 – 1943) 143
Евклид (III век до Р. Х.) 150
Клини С. К. (S. C. Kleene) 157, 158
Марков А. А. (младший, 1903 – 1979) 128
Матиясевич Ю. В. 15, 143
Мучни́к Альберт А. 91
Новиков П. С. (1901 – 1975) 128
Пост Э. (Emil L. Post, 1897 – 1954) 6, 13, 22, 91, 128, 157
Райс (Rice H. G.) 109
Роджерс Х. (H. Rogers, Jr.) 39, 58
Сипсер М. (Michael Sipser) 60
Тарский А. (A. Tarski, 1902 – 1983) 139, 140
Тьюринг А. М. (Alan Matison Turing, 1912 – 1954) 6, 77, 112, 153, 156, 157
Успенский В. А. 109
Ферма П. (P. Fermat (1601 – 1665) 15
Фридберг Р. (R. M. Friedberg) 91
Чёрч А. (A. Church) 157

КНИГИ ИЗДАТЕЛЬСТВА МЦНМО

Программирование: теоремы и задачи (А.Шень)

В этой книге объясняются приёмы, помогающие написать правильно и быстро работающую программу вместе с доказательством её правильности. Книга написана в форме задач с решениями.

Названия глав: 1. Переменные, выражения присваивания; 2. Порождение комбинаторных объектов; 3. Обход дерева. Перебор с возвратами; 4. Сортировка; 5. Конечные автоматы и обработка текстов; 6. Типы данных; 7. Рекурсия; 8. Как обойтись без рекурсии; 9. Разные алгоритмы на графах; 10. Сопоставление с образцом; 11. Представление множеств. Хеширование; 12. Представление множеств. Деревья. Сбалансированные деревья; 13. Контекстно-свободные грамматики; 14. Синтаксический разбор слева направо.

Издана в 1995 году. 264 с. Свободно распространяемый текст (ASCII, TEX, PostScript) находится по адресу

`ftp://ftp.mccme.ru/pub/users/shen/progbook`

Начала теории множеств (Н.К.Верещагин, А.Шень)

Основные понятия теории множеств (мощности, ординалы, трансфинитная индукция) входят в число вещей, которые хорошо бы знать каждому грамотному математику.

Книга представляет собой популярный рассказ об этих понятиях и основана на материалах лекций для младшекурсников, которые авторы читали на мехмате МГУ в разные годы.

Исходные тексты книги (TEX, PostScript) свободно распространяются и доступны по адресу

`ftp://ftp.mccme.ru/pub/users/shen/logic/sets`

Квантовые вычисления (М.Вялый, А.Китаев, А.Шень)

В 1997/8 учебном году А. Китаев прочёл в Независимом Московском университете спецкурс про квантовые вычисления, предварённый несколькими вводными лекциями про основы классической теории сложности вычислений (их прочёл А. Шень). На основе записей этого курса М. Вялый написал текст книги, которая представляет собой одно из первых в России и в мире систематических изложений основ теории квантовых вычислений.

Исходные тексты книги (TEX, PostScript) свободно распространяются и доступны по адресу

`ftp://ftp.mccme.ru/pub/users/shen/quantum`

Алгоритмы: построение и анализ

Книга представляет собой стандартный американский учебник по методам построения и анализа эффективных алгоритмов, написанный на основе одноимённого курса в Массачусетском технологическом институте и выдержавший более 15 изданий в США (первое — в 1990 году). Её авторы — Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest.

В ней разбираются важнейшие классы эффективных алгоритмов и методы их построения (как классические, так и современные). Изложение подробное (в ней около тысячи страниц, более 200 рисунков), понятное и математически строгое. Книгу можно использовать в качестве учебника или справочника, она будет полезна и студентам, и работающим программистам. Очень рекомендуем!

Как писал американский математик Эмиль Пост, понятие вычислимости может сыграть в истории дискретной математики роль, уступающую по значению лишь понятию натурального числа.

Теория вычислимых функций представляет интерес не только в связи с компьютерами; она является одной из центральных областей математической логики.

Многое из этой теории вполне доступно начинающим; мы пытались рассказать о ней, имея в виду самых разных читателей — от подготовленных школьников до профессиональных математиков.

Книга основана на материалах лекций для младшекурсников, которые авторы читали на мехмате МГУ в разные годы. В этой серии уже вышла книга «Начала теории множеств»; готовится к печати книга «Языки и исчисления».