

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ЯРОСЛАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМ. П.Г. ДЕМИДОВА

В.Г. ДУРНЕВ

ЭЛЕМЕНТЫ ТЕОРИИ АЛГОРИТМОВ

УЧЕБНОЕ ПОСОБИЕ

*Рекомендовано Учебно-методическим советом по математике и механике
Учебно-методического объединения
по классическому университетскому образованию РФ
для математических специальностей и направлений подготовки университетов*

ЯРОСЛАВЛЬ 2008

УДК 51:37
ББК В 12 я 73
Д 84

Рекомендовано
Редакционно-издательским советом университета
в качестве учебного издания. План 2008 года

Рецензенты:

кафедра алгебры и геометрии Тульского государственного
педагогического университета им. Л.Н. Толстого;
доктор физ.-матем. наук, профессор С.П. Струнков;
доктор физ.-матем. наук, профессор В.Н. Безверхний

Д 84 **Дурнев, В.Г.** Элементы теории алгоритмов: учеб. пособие /
В.Г. Дурнев; Яросл. гос. ун-т. — Ярославль: ЯрГУ, 2008. — 248 с.

ISBN 978-5-8397-0622-4

В учебном пособии рассматриваются основные понятия теории алгоритмов: машины Тьюринга, примитивно рекурсивные, рекурсивные и частично рекурсивные функции, рекурсивные и рекурсивно перечислимые множества, их нумерация, арифметизация теории машин Тьюринга, алгоритмически неразрешимые проблемы из теории алгоритмов, математической логики и алгебры, недетерминированные машины Тьюринга и классы \mathbb{NP} и \mathbb{P} .

Пособие предназначено для студентов, обучающихся по направлениям 010100 Математика и 010500 Прикладная математика и информатика, специальности 090102 Компьютерная безопасность, очной формы обучения. Оно может быть использовано при изучении дисциплин “Математическая логика и теория алгоритмов”, “Теория алгоритмов”, “Математическая логика” и “Дискретная математика и математическая логика” (блок ОПД, ДС), а также специальных дисциплин.

Библиогр.: 138 назв.

УДК 51:37
ББК В 12 я 73

ISBN 978-5-8397-0622-4

© Ярославский государственный университет
им. П.Г. Демидова, 2008

© В.Г. Дурнев, 2008

ОГЛАВЛЕНИЕ

Предисловие	5
Введение	9
ГЛАВА I. КОНЕЧНЫЕ АВТОМАТЫ И ЯЗЫКИ	21
§ 1. Алфавиты. Слова	21
§ 2. Конечные представления языков	25
§ 3. Конечные автоматы	27
§ 4. Конечные автоматы и регулярные выражения	35
§ 5. Некоторые алгоритмические проблемы для языков	41
§ 6. Минимизация автоматов	44
§ 7. Конечные автоматы с выходом	51
ГЛАВА II. ЧАСТИЧНО РЕКУРСИВНЫЕ ФУНКЦИИ	67
§ 1. Прimitивно рекурсивные и частично рекурсивные функции	67
§ 2. Нумерационные функции	82
§ 3. Рекурсивно перечислимые множества и предикаты	84
ГЛАВА III. МАШИНЫ ТЬЮРИНГА	95
§ 1. Машины Тьюринга	95
§ 2. Арифметизация теории машин Тьюринга	110
§ 3. Нумерация функций и множеств	119
ГЛАВА IV. АЛГОРИТМИЧЕСКИЕ ПРОБЛЕМЫ	129
§ 1. Арифметические множества и теорема А. Тарского	129
§ 2. Проблема выводимости для полусистем Туэ и комбинаторная проблема Поста	141
§ 3. Недетерминированные многоленточные машины Тьюринга	161
ГЛАВА V. \mathcal{NP} -ТРУДНЫЕ И \mathcal{NP} -ПОЛНЫЕ ПРОБЛЕМЫ	179
§ 1. \mathcal{NP} -полнота проблемы выполнимости для формул логики высказываний	179
§ 2. Сложность решения систем линейных уравнений	187
§ 3. \mathcal{NP} -полнота проблемы разрешимости для уравнений с нетривиальной правой частью в свободной полугруппе	208
§ 4. \mathcal{NP} -полные проблемы в теории графов	214
§ 5. Алгоритмически неразрешимые проблемы в области защиты информации	221
§ 6. Несколько NP -полных проблем	229
Послесловие	237
Литература	238

ПРЕДИСЛОВИЕ

Теория алгоритмов одновременно относится к числу древнейших и весьма молодых направлений математических исследований. Хотя первые примеры алгоритмов (вычислительных процессов механического характера) восходят к Древнему Египту, Вавилону и Греции, собственно теория алгоритмов ведет свое начало с 30-х годов XX века. Возникновение и развитие теории алгоритмов теснейшим образом было связано с развитием математической логики. XX век был периодом бурного развития математической логики, в рамках которой аксиоматический метод изложения математических теорий, ведущий свое начало от древнегреческих математиков, был доведен до своего логического завершения — были формализованы, выражены на подходящем формальном языке, не только основные положения ряда математических теорий, но и логические средства доказательства теорем в этих теориях. Именно в математической логике в 30-е годы XX века было выработано математическое уточнение интуитивного понятия алгоритма, которым к тому времени математики пользовались уже более двух тысяч лет. Это позволило установить существование неразрешимых алгоритмических проблем во многих разделах математики. В настоящее время математическая логика и теория алгоритмов имеют свой предмет и методы исследования, играют важную роль как в различных разделах математики, так и в ее приложениях, в частности в информатике. По мнению выдающегося специалиста в этой области, А.И. Мальцева, математическая логика и теория алгоритмов "образуют теоретический фундамент для создания и применения быстродействующих вычислительных и управляющих систем. Резко возрос удельный вес математической логики и теории алгоритмов и в самой математике" [37]. Это же подчеркивает и другой крупнейший специалист в области математической логики и теории алгоритмов Ю.Л. Ершов: "Специалисты в области ЭВМ начинают осознавать, что эти разделы математики являются фундаментом для построения действительно необходимой сейчас хорошей теории вычислений" [89].

При работе над пособием автором преследовалась цель — попытаться изложить с единых позиций основной, по его мнению, материал по теории алгоритмов, необходимый и в тоже время в идейном отношении доступный студентам III–IV курсов математических факультетов университетов. О некоторых особенностях отбора материала и выбранного нами способа его изложения будет сказано ниже. Заметим лишь, что автор больше заботился о ясности и полноте, чем о краткости изложения — доказательства некоторых теорем, конечно, можно сократить, однако при этом, по мнению автора, может пострадать ясность и несколько затрудниться понимание смысла проводимых построений. А ведь понимание доказательства включает в себя, прежде всего, понимание общей идеи доказательства, которая реализуется через этапы доказательства. Начинающему изучать предмет часто бывает проще понять длинное доказательство, в котором легко просматривается мотивировка каждого шага, чем краткое доказательство, основанное на оригинальной идее. В последнем случае

возникает вопрос "Почему так?". Конечно, после неоднократного использования оригинальная когда-то идея становится рабочим инструментом, и вопрос отпадает сам собой — эта идея становится частью нашего общего образования. Но, по нашему мнению, надо стремиться максимально облегчить труд начинающего читателя, а не ошеломлять его "феейверками мысли". Насколько это нам удалось — судить читателю. Автор сделал, что мог, пусть другие сделают больше.

При написании пособия использовались включенные в список литературы работы различных авторов на эту тему. Всем им, как и тем, чьи работы не включены в список литературы, однако оказали идейное влияние на формирование взглядов автора на предмет, мы выражаем искреннюю благодарность и признательность. Включенный в пособие материал можно считать уже достаточно устоявшимся, ставшим общематематическим достоянием. Хотя время от времени и появляются оригинальные доказательства известных в этой области теорем.

О содержании пособия можно судить по его достаточно подробному оглавлению. Приведем краткий обзор глав пособия.

Введение посвящено рассмотрению нескольких примеров алгоритмических проблем, исследование которых во многом способствовало формированию в начале XX века точного математического понятия алгоритма, предложенного в качестве эквивалента интуитивному понятию алгоритма.

Первая глава "Конечные автоматы и языки" служит определенным введением в теорию алгоритмов. Здесь определяются некоторые основные для дальнейшего изложения понятия — алфавиты, языки, регулярные выражения и регулярные языки, конечные детерминированные и недетерминированные автоматы, связь между ними. Центральной в этой главе является теорема С. Клини об эквивалентности понятий регулярного языка и конечно-автоматного языка. Рассмотрены некоторые алгоритмические проблемы для автоматных языков. Завершается глава рассмотрением некоторых вопросов, относящихся к конечным автоматам с выходом.

Вторая глава "Частично рекурсивные функции" посвящена рассмотрению примитивно рекурсивных, рекурсивных и частично рекурсивных функций — даются основные определения, устанавливается примитивная рекурсивность, рекурсивность и частичная рекурсивность ряда важных для дальнейшего изложения теоретико-числовых функций. Вводятся нумерационные функции, функция Геделя. Завершается глава рассмотрением рекурсивных и рекурсивно перечислимых множеств и предикатов.

В третьей главе "Машины Тьюринга" рассматриваются основные понятия, относящиеся к исторически одному из первых математических уточнений интуитивного понятия алгоритма — машине Тьюринга. Все вводимые понятия и доказываемые утверждения направлены на доказательство основного утверждения этой главы о совпадении класса частично рекурсивных функций с классом вычислимых по Тьюрингу функций. Завершается глава рассмотрением некоторых вопросов, связанных с нумерацией частично рекурсивных функций и

рекурсивно перечислимых множеств.

Четвертая глава "Алгоритмические проблемы" посвящена доказательству классической теоремы А. Тарского о неарифметичности понятия истинного арифметического утверждения. В этой главе в качестве приложения результатов, включенных в третью главу, устанавливается алгоритмическая неразрешимость проблемы выводимости для полусистем и систем Туэ — теорема А.А. Маркова — Э. Поста. Завершается глава обсуждением понятия недетерминированной многоленточной машины Тьюринга.

В пятой главе " \mathcal{NP} -трудные и \mathcal{NP} -полные проблемы" содержится некоторый достаточно традиционный материал, относящийся к этой тематике — доказательство теоремы С. Кука об \mathcal{NP} -полноте проблемы выполнимости для формул логики высказываний, сложность вопроса о разрешимости в натуральных числах систем линейных уравнений и одного уравнения, \mathcal{NP} -полные проблемы, относящиеся к вопросу о разрешимости уравнений весьма специального вида в свободной полугруппе. С целью демонстрации возможностей метода полиномиальной сводимости устанавливается \mathcal{NP} -полнота нескольких проблем из теории графов, путем полиномиального сведения одних проблем к другим. При этом в качестве исходной \mathcal{NP} -полной проблемы выступает проблема 3-выполнимости. Завершается глава небольшим списком \mathcal{NP} -полных проблем, идейно связанных с материалом пособия.

Пользуясь случаем, выражаю глубокую благодарность редактору пособия Марии Викторовне Никулиной за терпеливое и высоко качественное редактирование и Михаилу Анатольевичу Башкину за неоценимую помощь в компьютерном наборе и придании пособию достойного внешнего вида.

Введение

По мнению ряда специалистов по истории математики, возникновение различных вычислительных процедур, как говорят чисто механического характера, относится к самому раннему периоду развития математики – ко временам Древнего Египта, Вавилона и Греции. Эти процедуры позволяли находить величины, являвшиеся решениями некоторых задач, из величин, входящих в исходные данные этих задач, по достаточно точно сформулированным правилам (инструкциям). В качестве хорошо известных из школьного курса математики примеров можно привести формулы для нахождения площадей прямоугольников, параллелограммов, трапеций, треугольников, многоугольников и кругов, формулы для нахождения объемов параллелепипедов, призм, пирамид, конусов и шаров. Эти формулы позволяют по некоторым исходным данным находить соответствующие величины с помощью операций сложения, вычитания, умножения и деления. Формулы, выражающие корни алгебраических уравнений через их коэффициенты с помощью тех же операций сложения, вычитания, умножения, деления и извлечения корней, дают вычислительную процедуру для нахождения корней, по крайней мере, в принципе. Решением задачи на построение циркулем и линейкой до начала XIX века выступала соответствующая вычислительная процедура, и лишь в XIX веке решением некоторых таких задач стало доказательство невозможности выполнения тех или иных построений, т.е. невозможности создания соответствующей процедуры. Одним из наиболее известных таких вычислительных процессов является известный еще со времен Евклида процесс нахождения наибольшей общей меры двух отрезков, теперь более известный как алгоритм Евклида нахождения наибольшего общего делителя двух натуральных чисел и наибольшего общего делителя двух многочленов. Позже подобные вычислительные процедуры стали называться **алгоритмами** (**алгоритмами**). Другие примеры алгоритмов дают хорошо известные алгоритмы сложения и умножения натуральных чисел, заданных в некоторой системе счисления, в частности, в десятичной или двоичной. Считается, что само слово "**алгоритм**" (**алгоритм**) происходит от имени выдающегося узбекского математика IX столетия Мохаммеда ибн аль-Хорезми.

Аль-Хорезми Мохаммед бен Мусса (787 – около 850) – среднеазиатский математик, астроном и географ. Родом из Хорезмского оазиса, исторического центра цивилизации, которая дала миру целое созвездие выдающихся ученых и поэтов. С конца VIII века жил в Багдаде, где с 815 года был главой "Дома мудрости" – Багдадского хранилища рукописей.

Из математических работ аль-Хорезми до наших дней дошли два трактата – арифметический и алгебраический. Основные разделы первого трактата посвящены действиям с натуральными числами и дробями.

Во втором трактате изучаются уравнения первой и второй степени, рассматриваются некоторые приемы решения геометрических задач методами алгебры и ряд других вопросов.

Сочинения аль-Хорезми оказали большое влияние на развитие математики

не только на Востоке, но и в Западной Европе.

Считается, что от латинизированного имени аль-Хорезми (Algorithmi) происходит само слово "**алгоритм** (**алгоритм**)", которое еще в конце XIX – начале XX в. произносилось как "*алгоритм*", впрочем и в настоящее время некоторые математики придерживаются произношения "**алгоритм**", а не "**алгоритм**", например, "**нормальный алгоритм**" А.А. Маркова.

Напомним, что слово "*алгебра*" происходит от первого слова "*аль-джебр*" (восстановление) названия алгебраического трактата аль-Хорезми "Китаб аль-джебр валь-мукабала" ("Книга о восстановлении и противопоставлении"). В этом трактате *алгебра* впервые представлена как самостоятельный раздел математики, введены правила действий с алгебраическими величинами, решаются уравнения первой и второй степени. Латинский перевод XII века этого трактата многие годы в Европе был основным руководством по алгебре. Само слово "аль-джебр" обозначает операцию перенесения членов уравнения из одной его части в другую с изменением знака.

В книге "Об индийском счете" аль-Хорезми ввел десятичную систему счисления. Этот трактат в XII веке был переведен с арабского на латинский язык. Благодаря этому в Европе стала известна индийская позиционная система счисления и правила (алгоритмы) выполнения арифметических операций. Первоначально слово **алгоритм** (Algorithmi) служило обозначением правил выполнения действий в арифметике натуральных чисел, заданных в индийской позиционной системе счисления, а позже стало общим названием для всякой системы вычислений по строго определенным правилам. В средневековых манускриптах формула DIXIT ALGORIZMI – латинский перевод арабского QALA AL-KHOREZMI, по-русски ТАК СКАЗАЛ АЛЬ-ХОРЕЗМИ – служила сертификатом безупречной ясности и достоверности. Долгие годы аль-Хорезми считался высшим авторитетом.

Вся более чем четырехтысячелетняя история математики свидетельствует об особом интересе к алгоритмам, так как создание алгоритма для решения определенного бесконечного класса задач позволяет единообразно решить, по крайней мере, в принципе, любую из этих задач, и тем самым в определенной степени тривиализировать некоторую область математики. Можно с достаточным основанием утверждать, что значительный прогресс в развитии алгоритмических методов создал дошедшее до наших дней представление, что решение многих проблем должно носить алгоритмический характер. Яркими представителями таких взглядов были, например, Р. Декарт, Г.В. Лейбниц и Д. Гильберт. И они весьма преуспели в реализации своих научных взглядов. Создав аналитическую геометрию, Р. Декарт свел значительную часть геометрии к алгебраическим вычислениям и тем самым существенно продвинулся на пути алгоритмизации геометрии. Д. Гильберт в конце XIX века в определенной степени завершил работу по аксиоматизации евклидовой геометрии, что вместе с методами аналитической геометрии дало возможность сводить решение многих геометрических задач к вопросу об истинности определенных формул на множестве действительных чисел. А. Тарский разработал алгоритм, решаю-

ший последний вопрос, что явилось одним из первых примеров создания алгоритма для решения непростой математической задачи, для которой сам факт существования разрешающего алгоритма неочевиден.

Однако в середине 30-х годов XX века вера в определенную универсальность алгоритмических методов была подорвана.

Вплоть до 30-х годов XX века математики обходились неточным интуитивным понятием алгоритма. Возникавшие алгоритмические проблемы решались указанием соответствующих разрешающих процедур. При этом каждый раз, когда конкретный алгоритм для решения той или иной серии однотипных задач был построен, ни у кого не возникало сомнений в том, что указанная процедура является алгоритмом.

Под *алгоритмом в интуитивном смысле* мы понимаем точное предписание, определяющее вычислительный процесс, который ведет от исходных данных, варьируемых в некотором заданном множестве, к искомому результату. Этот вычислительный процесс должен быть конечным, он должен однозначно определяться заданием предписания и конкретного исходного данного. Иногда называют алгоритмом и сам вычислительный процесс, определяемый точным предписанием.

Под *алгоритмической проблемой* понимается задача построения алгоритма для решения заданной массовой задачи, т.е. бесконечной серии однотипных вопросов, зависящих от каких-то параметров. В случае, когда искомый алгоритм невозможен, говорят, что данная *алгоритмическая проблема неразрешима*. Неразрешимость алгоритмической проблемы, конечно, вовсе не означает, что какая-то из задач рассматриваемой серии неразрешима. Это означает лишь, что нельзя указать единый метод решения всех задач данной серии. Возможность решения каждой из них своим способом при этом не исключается.

Предписание, задающее алгоритм, должно быть *конечным*, и искомый результат должен получаться через *конечное число шагов* работы алгоритма. Оно должно быть настолько четким, чтобы однозначно определять весь вычислительный процесс от начальных данных до результата. Это означает, что процесс осуществляется *вычислителем* (будь то человек или машина) чисто механически, т.е. без привлечения каких-либо творческих элементов, и может быть воспроизведен с тем же результатом другим вычислителем и в другое время. Иначе говоря, для выполнения вычислительного процесса не требуется никакого творческого усилия.

По словам академика С.И. Адяна, к началу XX века математика стала богаче, в ней все большую роль стало играть изучение нечисловых объектов. *Слова в различных конечных или счетных алфавитах* все больше выступали в качестве данных для алгоритмов. Были явно сформулированы более глубокие алгоритмические проблемы, которые не удавалось решить, несмотря на усилия многих математиков. Начали возникать сомнения в возможности их положительного решения.

Приведем примеры таких проблем. Это прежде всего *проблема вывода*

мости для Исчисления Предикатов, состоящая в следующем:

для фиксированной конечной сигнатуры τ требуется разработать алгоритм, позволяющий по произвольной формуле Φ языка L_τ определить, выводима ли она в Исчислении Предикатов этой сигнатуры.

Эту проблему, получившую название *Entscheidungsproblem*, Д. Гильберт считал одной из основных проблем математики начала XX века. Заметим, что аналогичная **проблема выводимости для Исчисления Высказываний** решается путем построения истинностной таблицы для формулы Φ , так как по теореме Э. Поста формула Φ выводима в **Исчислении Высказываний** тогда и только тогда, когда она тождественно истинна в **Логике Высказываний**. Без особого успеха исследования по **проблеме выводимости для Исчисления Предикатов** велись с начала XX века до середины 30-х годов. И лишь в 1936 году А. Черчу удалось доказать невозможность построения алгоритма, решающего **проблему выводимости для Исчисления Предикатов**, но об этом речь пойдет позже.

Другая проблема такого рода – это **10-ая проблема Д. Гильберта**, включенная под номером 10 в знаменитый список из 23 проблем, сформулированных им на II Международном математическом конгрессе, состоявшемся в августе 1900 года в Париже:

10. Выяснение разрешимости произвольного диофантова уравнения. Пусть задано диофантово уравнение с произвольным числом неизвестных и целыми рациональными коэффициентами; требуется указать способ, по которому с помощью конечного числа операций можно было бы узнать, разрешимо ли уравнение в целых рациональных числах или нет.

Под диофантовым уравнением понимается уравнение вида

$$F(x_1, \dots, x_n) = 0,$$

где $F(x_1, \dots, x_n)$ – полином с целыми коэффициентами от переменных x_1, \dots, x_n . "Способ", о котором идет речь в формулировке 10-ой проблемы, теперь понимается как **алгоритм**. Хорошо известно, какие трудности возникают при исследовании диофантовых уравнений, даже такого, казалось бы простого, как уравнение Пелля $x^2 - dy^2 = 1$. Поэтому появились предположения, что искомого алгоритма просто не существует. Справедливость этих предположений была установлена во второй половине XX века в серии работ М. Дэвиса, Х. Путнам, Дж. Робинсон и Ю.В. Матиясевича [48], что явилось одним из выдающихся фундаментальных достижений теории алгоритмов второй половины XX века. Заметим, что в настоящее время неизвестно, возможно ли построить алгоритм, позволяющий по произвольному уравнению вида

$$F(x_1, \dots, x_n) = 0,$$

где $F(x_1, \dots, x_n)$ – полином с целыми коэффициентами от переменных x_1, \dots, x_n , определить, имеет ли оно решение в рациональных числах. Интересно это сопоставить со следующим фактом: вопрос о разрешимости уравнений указанного

вида в действительных числах алгоритмически разрешим, что было установлено А. Тарским на основе глубокого обобщения известного читателю из курса "Алгебра" метода Штурма, относящегося к уравнениям с одной неизвестной, т.е. когда $n = 1$, а вопрос о разрешимости уравнений этого вида в комплексных числах легко решается на основе так называемой основной теоремы о многочленах над полем комплексных чисел.

Приведем еще один пример алгоритмической проблемы, долгие годы не поддававшейся решению. Речь пойдет о **проблеме выводимости** для полусистем Туэ, сформулированной норвежским математиком Акселем Туэ в 1914 году.

Пусть $\mathcal{A} = \{a_1, \dots, a_n\}$ – произвольный конечный алфавит. Зафиксируем некоторый конечный набор упорядоченных пар слов $\langle A_1, B_1 \rangle, \dots, \langle A_m, B_m \rangle$ в этом алфавите. С каждой парой $\langle A_i, B_i \rangle$ свяжем *элементарное преобразование* слов в алфавите \mathcal{A} – переход вида

$$UA_iV \longrightarrow UB_iV,$$

где U и V – произвольные слова в алфавите \mathcal{A} . Саму упорядоченную пару слов $\langle A_i, B_i \rangle$ традиционно обозначают в виде $A_i \rightarrow B_i$. Полученный объект обозначается в виде

$$\langle a_1, \dots, a_n \mid A_1 \rightarrow B_1, \dots, A_m \rightarrow B_m \rangle$$

и называется **полусистемой Туэ** или *системой полу-Туэ* и будет обозначаться через ST . При этом $\mathcal{A} = \{a_1, \dots, a_n\}$ называется *алфавитом полусистемы ST* , а $A_1 \rightarrow B_1, \dots, A_m \rightarrow B_m$ – ее *системой подстановок*.

Основным понятием для полусистем Туэ является **понятие выводимости**.

Пусть W и U – слова в алфавите полусистемы Туэ

$$ST = \langle a_1, \dots, a_n \mid A_1 \rightarrow B_1, \dots, A_m \rightarrow B_m \rangle.$$

Слово U называется **выводимым** из слова W , если существует последовательность элементарных преобразований

$$W = W_0 \rightarrow W_1 \rightarrow \dots \rightarrow W_k \rightarrow W_{k+1} \rightarrow \dots \rightarrow W_s = U,$$

переводящая слово U в слово W .

В **проблеме выводимости для полусистем Туэ**, сформулированной А. Туэ в 1914 году, требуется разработать общий метод, позволяющий по любым двум словам W и U в алфавите полусистемы Туэ ST определить, выводимо ли слово U из слова W .

Аксель Туэ (19.02.1863 – 7.03.1922) – норвежский математик. Ему принадлежат важные результаты в теории чисел, в диофантовом анализе и в разработке метода тригонометрических сумм. Широкую известность получили метод Туэ в теории диофантовых приближений, теорема Туэ – Зигеля – Рота, теорема Туэ о конечности числа целочисленных решений однородного диофантова уравнения с двумя неизвестными и системы Туэ.

Следующий пример относится к теории групп. Он связан с введенным Вальтером фон Диком в 1882 – 1883 годах способом *задания групп образующими элементами и определяющими соотношениями*. Такой способ задания групп возникает естественным образом в топологии как способ задания фундаментальных групп некоторых топологических пространств.

Пусть $\mathcal{A} = \{a_1, \dots, a_n\}$ – произвольный конечный алфавит. Введем алфавит букв-двойников $\mathcal{A}^{-1} = \{a_1^{-1}, \dots, a_n^{-1}\}$. Объединение $\mathcal{A} \cup \mathcal{A}^{-1}$ этих алфавитов будем называть *групповым алфавитом*.

Зафиксируем некоторый конечный набор упорядоченных пар слов $\langle A_1, B_1 \rangle, \dots, \langle A_m, B_m \rangle$ в этом групповом алфавите. С каждой парой $\langle A_i, B_i \rangle$ свяжем два *элементарных преобразования* слов в групповом алфавите $\mathcal{A} \cup \mathcal{A}^{-1}$ – переходы вида

$$UA_iV \longrightarrow UB_iV, \quad UB_iV \longrightarrow UA_iV,$$

где U и V – произвольные слова в групповом алфавите.

К этим элементарным преобразованиям добавим так называемые *тривиальные элементарные преобразования* слов в групповом алфавите $\mathcal{A} \cup \mathcal{A}^{-1}$ – переходы вида

$$Ua_i^\varepsilon a_i^{-\varepsilon}V \longrightarrow UV, \quad UV \longrightarrow Ua_i^\varepsilon a_i^{-\varepsilon}V,$$

где U и V – произвольные слова в групповом алфавите, а $\varepsilon \in \{-1, 1\}$. Преобразования первого вида называются *сокращениями*, а второго – *вставками*.

Саму упорядоченную пару слов $\langle A_i, B_i \rangle$ традиционно обозначают в виде $A_i = B_i$. Полученный объект обозначается в виде

$$\langle\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle\rangle,$$

и он будет служить заданием некоторой группы, которая будет обозначаться тем же способом. При этом a_1, \dots, a_n называются *образующими элементами* этой группы, а $A_1 = B_1, \dots, A_m = B_m$ – ее *определяющими соотношениями*.

Для построения этой группы используем отношение выводимости слов.

Как и в случае полусистем Туэ, слово U называется *выводимым* из слова W (обозначается $W \vdash^* U$), если существует последовательность элементарных преобразований

$$W = W_0 \rightarrow W_1 \rightarrow \dots \rightarrow W_k \rightarrow W_{k+1} \rightarrow \dots \rightarrow W_s = U,$$

переводящая слово U в слово W .

Нетрудно показать, что отношение выводимости \vdash^* в рассматриваемом случае является отношением эквивалентности, т.е. оно рефлексивно, транзитивно и симметрично. Соответствующие классы эквивалентности будем обозначать через $[W]$.

На множестве классов эквивалентности естественным образом определяется умножение равенством

$$[W] \cdot [U] = [WU],$$

где WU – обычное произведение (сочленение, конкатенация) слов W и U .

Нетрудно проверить, что множество классов эквивалентности относительно введенной операции умножения является группой. При этом роль нейтрального элемента выполняет класс $[1]$, где через 1 обозначено пустое слово, а элементом, обратным к $[W]$, является $[\overline{W}]$, где

$$\overline{a_{i_1}^{\varepsilon_1} \dots a_{i_t}^{\varepsilon_t}} = a_{i_t}^{-\varepsilon_t} \dots a_{i_1}^{-\varepsilon_1}.$$

Построенная группа обозначается через

$$\langle\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle\rangle$$

и называется *группой, заданной образующими элементами a_1, \dots, a_n и определяющими соотношениями $A_1 = B_1, \dots, A_m = B_m$* .

Если некоторая группа G изоморфна построенной группе, то говорят, что группа G имеет задание (*генетический код, копредставление*)

$$\langle\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle\rangle.$$

Например, симметрическая группа S_3 имеет задание

$$\langle\langle a, b \mid a^3 = 1, b^2 = 1, ba = a^2b \rangle\rangle.$$

Группа $SL(2, Z)$ целочисленных матриц второго порядка с определителем, равным единице, имеет задание

$$\langle\langle a, b \mid a^6 = 1, b^4 = 1, a^3 = b^2 \rangle\rangle,$$

а ее факторгруппа по центру $PSL(2, Z)$ (проективная специальная целочисленная группа матриц второго порядка) имеет задание

$$\langle\langle a, b \mid a^3 = 1, b^2 = 1 \rangle\rangle.$$

Заметим, что группа кос на трех нитях и группа узла клеверный лист (трилистник) имеют одно и то же задание

$$\langle\langle a, b \mid a^3 = b^2 \rangle\rangle.$$

В связи с рассмотренным способом задания групп М. Дэн в работе 1911 года сформулировал три алгоритмические проблемы, получившие название *фундаментальные проблемы М. Дэна* – **проблему тождества**, **проблему сопряженности** и **проблему изоморфизма**.

Проблема тождества. Требуется разработать общий метод (алгоритм), позволяющий по любому заданию группы

$$\langle\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle\rangle$$

и по любым двум (групповым) словам W и U в этих образующих определить, равны ли элементы $[W]$ и $[U]$, т.е. можно ли из слова W вывести слово U ,

пользуясь указанными определяющими соотношениями и тривиальными соотношениями.

Проблема сопряженности. Требуется разработать общий метод (алгоритм), позволяющий по любому заданию группы

$$\langle\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle\rangle$$

и по любым двум (групповым) словам W и U в этих образующих определить, сопряжены ли элементы $[W]$ и $[U]$, т.е. найдется ли такое слово Z , что $[Z]^{-1}[W][Z] = [U]$ (можно ли из слова $\bar{Z}WZ$ вывести слово U , пользуясь указанными определяющими соотношениями и тривиальными соотношениями).

Проблема изоморфизма. Требуется разработать общий метод (алгоритм), позволяющий по любым двум заданиям

$$\langle\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle\rangle$$

и

$$\langle\langle b_1, \dots, b_p \mid C_1 = D_1, \dots, C_q = D_q \rangle\rangle$$

определить, будут ли изоморфны соответствующие группы.

Первые две проблемы были сформулированы М. Дэном в предыдущей работе 1910 года, а третью проблему можно обнаружить в работе Х. Титце 1908 года, но не выделенную там специально. Однако особое внимание этим проблемам было уделено именно в работе М. Дэна 1911 года, которая начинается с формулировки этих трех проблем.

В большой работе 1895 года "Analysis Situs" А. Пуанкаре ввел понятие фундаментальной группы, а в столь же большой работе 1908 года Х. Титце установил, что фундаментальные группы некоторых многообразий, заданных клеточными комплексами, имеют конечные задания. Начиная с работ Ж. Листинга 1848 года интенсивно изучаемый класс наглядных топологических объектов составляли узлы. В докладе 1905 года В. Виртингер изложил метод нахождения группы узла по его проекции на евклидову плоскость. Ранее М. Дэн предложил несколько иной способ нахождения задания группы узла, однако впоследствии метод В. Виртингера стал более распространенным. М. Дэн доказал, что узел изотопически эквивалентен окружности тогда и только тогда, когда его группа абелева, а значит циклическая. Из сказанного можно понять, почему фундаментальные проблемы М. Дэна сразу привлекли внимание исследователей. Заметим лишь, что в проблемах М. Дэна речь шла о построении соответствующих разрешающих алгоритмов.

Почти полвека проблемы М. Дэна не поддавались решению – только в начале 50-х годов XX века Петр Сергеевич Новиков доказал, что *искомые алгоритмы невозможно построить*.

Так как проблемы М. Дэна решить не удавалось, то естественно возникло желание рассмотреть более общую ситуацию – аналогичные алгоритмические проблемы для полугрупп, заданных образующими элементами и определяющими соотношениями, и установить их неразрешимость.

В 40-е годы XX века А.А. Марков и Э. Пост независимо и практически одновременно установили алгоритмическую неразрешимость проблемы равенства (эквивалентности) слов для полугрупп, заданных конечным числом образующих элементов и конечным числом определяющих соотношений. Это понятие вводится аналогично введенному выше понятию группы, заданной образующими элементами и определяющими соотношениями.

Пусть $\mathcal{A} = \{a_1, \dots, a_n\}$ – произвольный конечный алфавит.

Зафиксируем некоторый конечный набор упорядоченных пар слов $\langle A_1, B_1 \rangle, \dots, \langle A_m, B_m \rangle$ в этом алфавите. С каждой парой $\langle A_i, B_i \rangle$ свяжем два *элементарных преобразования* слов в алфавите \mathcal{A} – переходы вида

$$UA_iV \longrightarrow UB_iV, \quad UB_iV \longrightarrow UA_iV,$$

где U и V – произвольные слова в этом алфавите.

Саму упорядоченную пару слов $\langle A_i, B_i \rangle$ традиционно обозначают в виде $A_i = B_i$. Полученный объект обозначается в виде

$$\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle,$$

и он будет служить заданием некоторой полугруппы, которая будет обозначаться тем же способом. При этом a_1, \dots, a_n называются *образующими элементами* этой полугруппы, а $A_1 = B_1, \dots, A_m = B_m$ – ее *определяющими соотношениями*.

Для построения этой полугруппы, как и в случае группы, используем отношение выводимости слов.

Слово U называется *выводимым* из слова W (обозначается $W \vdash^* U$), если либо оно ему графически равно (выводимо за 0 шагов), либо существует последовательность элементарных преобразований

$$W = W_0 \rightarrow W_1 \rightarrow \dots \rightarrow W_k \rightarrow W_{k+1} \rightarrow \dots \rightarrow W_s = U,$$

переводящая слово U в слово W .

Нетрудно показать, что отношение выводимости \vdash^* в рассматриваемом случае является отношением эквивалентности, т.е. оно рефлексивно, транзитивно и симметрично. Соответствующие классы эквивалентности будем обозначать через $[W]$.

На множестве классов эквивалентности естественным образом определяется умножение равенством

$$[W] \cdot [U] = [WU],$$

где WU – обычное произведение (сочленение, конкатенация) слов W и U .

Нетрудно проверить, что множество классов эквивалентности относительно введенной операции умножения является полугруппой с единицей (моноидом). При этом роль нейтрального элемента выполняет класс $[1]$, где через 1 обозначено пустое слово. Построенная полугруппа обозначается через

$$\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle$$

и называется *полугруппой*, заданной образующими элементами a_1, \dots, a_n и определяющими соотношениями $A_1 = B_1, \dots, A_m = B_m$.

Если некоторая полугруппа S изоморфна построенной полугруппе, то говорят, что *полугруппа S имеет задание (генетический код, копредставление)*

$$\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle.$$

В связи с рассмотренным способом задания полугрупп естественно возникают алгоритмические проблемы, аналогичные проблемам М. Дэна, – **проблема равенства (эквивалентности) слов для полугрупп** и **проблема изоморфизма для полугрупп**.

Проблема равенства (эквивалентности) слов для полугрупп. Требуется разработать общий метод (алгоритм), позволяющий по любому заданию полугруппы

$$\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle$$

и по любым двум словам W и U в этих образующих определить, равны ли элементы $[W]$ и $[U]$, т.е. можно ли из слова W вывести слово U , пользуясь указанными определяющими соотношениями.

Проблема изоморфизма для полугрупп. Требуется разработать общий метод (алгоритм), позволяющий по любым двум заданиям

$$\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle$$

и

$$\langle b_1, \dots, b_p \mid C_1 = D_1, \dots, C_q = D_q \rangle$$

определить, будут ли изоморфны соответствующие полугруппы.

Как было сказано выше, в 40-е годы XX века А.А. Марков и Э. Пост независимо и практически одновременно установили алгоритмическую неразрешимость проблемы равенства (эквивалентности) слов для полугрупп, заданных конечным числом образующих элементов и конечным числом определяющих соотношений. Неразрешимость проблемы изоморфизма для полугрупп легко следует из неразрешимости для них проблемы равенства слов.

Для доказательства невозможности алгоритма, дающего решение той или иной серии задач, следовало сначала точно математически определить, какой смысл вкладывается в понятие "алгоритм". Такое уточнение определения понятия алгоритма стало возможным благодаря развитию математической логики в начале XX века. Оно было получено в середине 1930-х годов в работах К. Геделя, Д. Эрбрана, А. Черча, Э. Поста и А. Тьюринга почти одновременно в двух внешне совершенно различных формах: в виде точного математического описания класса вычислимых функций натурального аргумента (**частично рекурсивные и рекурсивные функции**) и в виде точного математического определения класса вычислительных процессов, реализуемых механически абстрактным вычислителем (**машины Тьюринга**). Вскоре было установлено,

что эти два уточнения понятия алгоритма по существу эквивалентны друг другу, т.е. они взаимозаменяемы. Это дало основание американскому математику А. Черчу в 1936 году [102] сформулировать следующий тезис:

Тезис Черча. *Всякий алгоритм в интуитивном смысле с точки зрения его вычислительных возможностей эквивалентен некоторому алгоритму в уточненном смысле.*

Появление точного понятия алгоритма позволило установить неразрешимость многих алгоритмических проблем. Сначала неразрешимые алгоритмические проблемы были найдены в самой теории алгоритмов, затем в математической логике. Позже было установлено их существование и в других разделах математики – алгебре, топологии, теории чисел, анализе, теории обыкновенных дифференциальных уравнений. Было доказано, что они есть и среди известных задач, поставленных в математике ранее, до создания теории алгоритмов, и долгие годы не поддававшихся решению. Таковыми оказались в математической логике – **проблема выводимости для полусистем Туэ**, **проблема выводимости для Исчисления Предикатов**, в алгебре – **фундаментальные проблемы М. Дэна для конечно определенных групп** и аналогичные проблемы для конечно определенных полугрупп, в топологии – **проблема гомеоморфизма для полиэдров** (многообразий, являющихся ”правильными” объединениями со склейкой стандартных симплексов), в теории чисел – **проблема разрешимости в целых или натуральных числах полиномиальных уравнений $F(x_1, \dots, x_n) = 0$** (10-ая проблема Д. Гильберта). Был найден ряд алгоритмически неразрешимых проблем, связанных с вопросом о существовании решения для систем обыкновенных дифференциальных уравнений, с вопросом о сходимости несобственных интегралов и наличием первообразных из некоторого фиксированного класса для функций определенного класса [49].

В первой половине XX века исследователей прежде всего интересовало, существует ли алгоритм для решения рассматриваемой задачи. Во второй половине XX века особый интерес стал представлять вопрос о сложности соответствующего алгоритма, т.е. об используемых им ресурсах – времени и памяти. В определенной мере это было связано с компьютерной реализацией алгоритмов. Наиболее интересными представляются вопросы, связанные с получением нетривиальных нижних оценок сложности алгоритмов, решающих заданную задачу. Однако это тема выходит за рамки нашего пособия. Мы планируем ей посвятить в будущем некоторое дополнение к пособию.

ГЛАВА I

КОНЕЧНЫЕ АВТОМАТЫ И ЯЗЫКИ

§1. Алфавиты. Слова

При написании этого параграфа мы использовали электронные варианты конспектов лекций С.Л. Березнюка [6] и А.С. Морозова [54] как при отборе материала, так и при выборе способа его изложения.

В этом параграфе, как следует из его названия, будут рассматриваться **алфавиты** и **языки**. Как и многие другие понятия, например, понятие аксиомы, понятие языка и алфавита прошло длительный этап развития (становления, уточнения, формализации). На общий вопрос "Что такое язык?" так же трудно дать исчерпывающий ответ, как и на вопрос "Что такое число?" и на многочисленные аналогичные вопросы, касающиеся "слишком общих понятий". Многие языки не требуют слишком точных формулировок, а некоторым из них они просто вредны, например, некоторым поэтическим языкам и языкам живописи. Однако ряду языков требуется точность и определенная однозначность, например, языкам программирования, юридическим языкам и т.д. и т.п.

Мы будем рассматривать лишь языки, в которых на сегодняшний день достигнут высший по современным представлениям уровень точности формулировок и определений, – **формальные языки**. К ним относятся компьютерные языки, языки программирования и математические языки, в первую очередь, – язык математической логики. Под понятием языка мы будем понимать нечто может быть узкое, однако поддающееся точному математическому определению, а значит и изучению.

Определение 1.1. *Алфавитом называется любое непустое множество Σ символов. Элементы множества Σ называются **символами** или **буквами** данного алфавита.*

Никаких ограничений на природу элементов множества Σ (символов алфавита) мы не накладываем, единственное предположение относительно Σ состоит

в том, что мы должны уметь распознавать каждый символ из Σ как тот же самый при каждом из его вхождений и отличать его от всех других символов из Σ , т.е. предполагается, что **различия неодинаковых символов значительно превосходят мелкие различия одинаковых символов**. Символы из Σ рассматриваются как простые знаки, а не как символы, которые что-либо означают. Символы алфавита Σ считаются *элементарными знаками*, далее неделимыми. Например, если W и V входят в Σ , то мы считаем, что W – это элементарный объект, далее неделимый, в частности, не считаем, что W – это два раза повторенное V .

Определение 1.2. *Словом или выражением в алфавите Σ называется произвольная конечная (возможно пустая) последовательность*

$$w_1 w_2 \dots w_n$$

*символов из Σ , т.е. $n \geq 0$ и при любом i ($1 \leq i \leq n$) $w_i \in \Sigma$. Число n называется **длиной** слова $w_1 w_2 \dots w_n$.*

Длину произвольного слова X будем обозначать через $|X|$. Пустое слово – это слово длины 0, будем его обозначать через Λ , ε или через 1, если это не приведет к путанице.

Множество всех слов в алфавите Σ традиционно обозначается через Σ^* .

Пусть X – это слово $w_1 w_2 \dots w_n$ ($w_i \in \Sigma$, $i = 1, \dots, n$), а Y – это слово $v_1 v_2 \dots v_m$ ($v_i \in \Sigma$, $i = 1, \dots, m$).

Определение 1.3. *Слова X и Y называются **графически равными**, если $n = m$ и при любом i ($i = 1, \dots, n$) w_i и v_i один и тот же символ из Σ .*

Утверждение “слова X и Y равны графически” будем записывать в виде

$$X = Y.$$

Определение 1.4. *Произведением (соединением, сочленением, конкатенацией) слов X и Y называется слово XY , т.е. если X – это слово $w_1 w_2 \dots w_n$, а Y – это слово $v_1 v_2 \dots v_m$, то XY – это слово*

$$w_1 w_2 \dots w_n v_1 v_2 \dots v_m.$$

Иногда вместо обозначения XY будем использовать $X \circ Y$.

Очевидно, что $|XY| = |X| + |Y|$ и $X\Lambda = \Lambda X = X$.

Отметим следующее очевидное свойство: если $X = Y$, а Z – любое слово, то

$$XZ = YZ \quad ZX = ZY$$

и обратно, если

$$XZ = YZ \quad \text{или} \quad ZX = ZY,$$

то $X = Y$.

Определение 1.5. Слово X называется **подсловом** слова Y , если найдутся такие слова U и V , что выполняется равенство $Y = UXV$. При этом, если слово U пусто, т.е. выполняется равенство $Y = XV$, то слово X называется **началом** слова Y . Если же пусто слово V , т.е. выполняется равенство $Y = UX$, то слово X называется **концом** слова Y .

Лемма 1.1. Если X, Y, Z, V – слова в алфавите Σ и $XY = ZV$, то для некоторого слова P либо $X = ZP$ и $V = PY$, либо $Z = XP$ и $Y = PV$.

Д о к а з а т е л ь с т в о. Если $|Z| \leq |X|$, то, очевидно, найдется такое слово P (возможно пустое), что $X = ZP$, но тогда $ZPY = ZV$ и, значит, $V = PY$. В случае, когда $|X| \leq |Z|$, совершенно аналогично показываем, что для некоторого слова P

$$Z = XP \quad Y = PV.$$

Это завершает доказательство леммы. \square

Определение 1.6. Пусть $*$ $\notin \Sigma$. **Вхождением** слова Y в слово X называется слово V вида $X_1 * Y * X_2$ при условии, что $X = X_1 Y X_2$.

Определение 1.7. **Результатом замены данного вхождения** $V = X_1 * Y * X_2$ слова Y в слово X на слово Z называется слово $X_1 Z X_2$.

Ясно, что одно и то же слово Y может иметь более одного вхождения в слово X , поэтому следует говорить о *замене данного вхождения* V слова Y в слово X на слово Z , а не просто о замене слова Y в слове X на слово Z . В частности, если Y – пустое слово, то вхождение V слова Y в слово X имеет вид $X_1 ** X_2$, и тогда результат замены вхождения V слова Y в слово X на слово Z – это слово $X_1 Z X_2$.

Если слово Y состоит из одного символа, т.е. Y – это некоторое α ($\alpha \in \Sigma$), то вхождение V слова Y в слово X называется **вхождением буквы (символа)** α в слово X .

Если α – символ алфавита Σ , а X и Z – слова в алфавите Σ , то **результат одновременной замены каждого вхождения символа α в слово X на слово Z** обозначается через

$$X_\alpha[Z].$$

Аналогично, если $\alpha_1, \dots, \alpha_n$ – попарно различные символы из Σ , а X, Z_1, \dots, Z_n – слова в алфавите Σ , то **результат одновременной замены всех вхождений $\alpha_1, \dots, \alpha_n$ соответственно на слова Z_1, \dots, Z_n** называется **результатом одновременной подстановки** слов Z_1, \dots, Z_n вместо $\alpha_1, \dots, \alpha_n$ и обозначается через $X_{\alpha_1, \dots, \alpha_n}[Z_1, \dots, Z_n]$.

Для любого слова w и любого натурального числа n определим слово w^n следующим образом:

$$w^0 = e, \quad w^{n+1} = w^n w \text{ для каждого } n \geq 0.$$

Обращением слова w , обозначается через w^R , называется "то же самое слово, записанное справа налево". Приведем формальное определение.

Определение 1.8. 1) Если w – слово нулевой длины, то $w^R = w = \varepsilon$.

2) Если w – слово длины $n+1$ и $w = ua$ для некоторого $a \in \Sigma$, то $w^R = aw^R$.

Определение 1.9. Языком в алфавите Σ называется любое подмножество множества Σ^* всех слов в этом алфавите.

Примерами языков в алфавите Σ могут служить: Σ^* – универсальный язык, \emptyset – пустой язык и $\{a\}$ – однобуквенный язык, где $a \in \Sigma$.

Конечные языки можно задавать путем перечисления всех их элементов. Однако большинство интересных языков являются бесконечными, поэтому невозможно их задание перечислением всех элементов. Например, важную роль играют языки

$$\{w \mid w \in \Sigma^* \text{ \& } w = w^R\}, \quad \{0^n 1^n \mid n \geq 0\} \\ \{w \mid w \in \{0,1\}^* \text{ \& } w \text{ содержит одинаковое число нулей и единиц}\}.$$

Бесконечные языки обычно задаются в виде

$$L = \{w \mid w \in \Sigma^* \text{ \& } w \text{ обладает свойством } P\}.$$

Из теории множеств хорошо известно, что если Σ – конечный или счетный алфавит, то множество Σ^* всех слов в этом алфавите является счетным.

Языки – это множества, поэтому к ним можно применять теоретико-множественные операции объединения, пересечения и разности.

Однако существуют некоторые важные операции, которые имеют смысл именно для языков.

Это прежде всего **произведение** или **конкатенация** языков. Если L_1 и L_2 – языки в алфавите Σ , то их (произведение) конкатенация $L_1 \circ L_2$ задается равенством

$$L_1 \circ L_2 = \{w \mid w \in \Sigma^* \text{ \& для некоторых } x \in L_1 \text{ и } y \in L_2 \text{ } w = x \circ y\}.$$

Еще одна операция над языками – это **навешивание звездочки Клини** на язык L . Результат обозначается через L^* . L^* – это множество всех слов, получаемых путем конкатенации нуля или более слов из L (конкатенация нуля слов – это ε , а конкатенация одного слова – это само это слово).

$$L^* = \{w \mid w \in \Sigma^* \text{ \& } \\ w = w_1 \circ \dots \circ w_n \text{ для некоторого } n \geq 0 \text{ и } w_1, \dots, w_n \in L\}.$$

Через L^+ обозначается язык LL^* , т.е.

$$L^+ = \{w \mid w \in \Sigma^* \text{ \& } \\ w = w_1 \circ w_2 \circ \dots \circ w_n \text{ для некоторого } n \geq 1 \text{ и } w_1, \dots, w_n \in L\}.$$

§2. Конечные представления языков

Конечный язык может быть задан, в принципе, перечислением всех входящих в него слов. Рассмотрим вопрос о возможности конечного задания (описания) бесконечных языков.

Прежде всего необходимо уточнить, что мы понимаем под словами "конечное задание (описание) языка". Во-первых, каждое такое задание само будет словом в некотором алфавите Σ . Во-вторых, естественно требование, чтобы различные языки имели различные задания. Так как множество Σ^* всех слов в алфавите Σ счетно, то счетно и множество возможных конечных заданий языков. Однако множество всех возможных языков над данным алфавитом Σ несчетно. Поэтому не каждый язык допускает конечное задание. Следовательно, разумно попытаться найти некоторые конечные задания для достаточно интересных с разных точек зрения языков.

Один из подходов к решению этой задачи состоит в использовании аппарата **регулярных выражений**.

Фиксируем алфавит Σ . Дадим индуктивное определение понятия **регулярное выражение над алфавитом Σ и определяемого им языка**. Если **регулярное выражение** обозначено через α , то **определяемый им язык** будем обозначать через $L(\alpha)$.

- 1) \emptyset – регулярное выражение. $L(\emptyset)$ – пустой язык \emptyset .
 - 2) ε – регулярное выражение. $L(\varepsilon) = \{\varepsilon\}$ – язык, состоящий лишь из пустого слова ε .
 - 3) Если a – буква (символ) алфавита Σ , то a – регулярное выражение. $L(a) = \{a\}$ – язык, состоящий лишь из символа a .
- Пункты 1), 2) и 3) составляют базу индуктивного определения. Следующие три пункта – это индуктивный переход.
- 4) Если α и β – регулярные выражения, то $(\alpha\beta)$ – регулярное выражение и $L(\alpha\beta) = L(\alpha) \cdot L(\beta)$ – произведение языков $L(\alpha)$ и $L(\beta)$.
 - 5) Если α и β – регулярные выражения, то $(\alpha \cup \beta)$ – регулярное выражение и $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$ – объединение языков $L(\alpha)$ и $L(\beta)$.
 - 6) Если α – регулярное выражение, то α^* – регулярное выражение и $L(\alpha^*) = (L(\alpha))^*$.

Язык называется **регулярным**, если он может быть задан некоторым регулярным выражением.

Заметим, что разные регулярные выражения могут задавать один и тот же язык. Более того, один и тот же регулярный язык можно задать бесконечным множеством регулярных выражений.

Легко понять, что класс регулярных языков в алфавите Σ замкнут относительно операций \cup , \cdot и $*$. Позже будет установлена его замкнутость и относительно операций \cap и \setminus .

Существенную часть программного обеспечения современных компьютеров составляют компиляторы. С одной стороны, это связано с тем, что в настоящее время языки высокого уровня – основное средство разработки программ, лишь

незначительная часть программного обеспечения, требующая особой эффективности и надежности, программируется с помощью ассемблеров. С другой стороны, бурное развитие архитектур ЭВМ также поддерживает потребность в создании новых компиляторов. Важными этапами процесса компиляции являются *лексический и синтаксический анализ*. На этапе лексического анализа текст входной программы, являющийся просто словом в соответствующем алфавите языка программирования, на котором написана программа, разбивается в соответствии с определениями используемого языка программирования на лексемы. В основу реализации лексических анализаторов часто закладывается формализм регулярных выражений и конечных автоматов. Работа лексического анализатора осуществляется некоторым конечным автоматом, распознающим язык лексем. Но с практической точки зрения не всегда удобно непосредственно задавать соответствующий автомат. Вместо этого используется формализм регулярных выражений или праволинейных грамматик, по которым уже можно построить соответствующий конечный автомат.

Рассмотрим пример описания лексических структур с помощью регулярных выражений. При этом удобно сопоставлять регулярным выражениям их имена, а затем делать ссылки на лексемы по их именам. Для определения имен лексем используют выражения вида, которые иногда называются регулярными определениями

$$I_1 = \alpha_1, I_2 = \alpha_2, \dots, I_m = \alpha_m,$$

где I_1, I_2, \dots, I_m – различные имена, а при любом $0 \leq t < m$ α_{t+1} – регулярное выражение в алфавите $\Sigma \cup \{I_1, I_2, \dots, I_t\}$, где Σ – основной алфавит. Заменяя имена регулярных выражений на соответствующие им регулярные выражения, можно для любого имени I_t построить соответствующее регулярное выражение в основном алфавите Σ .

Введем общепринятое соглашение о возможности опускать ”лишние” скобки и использовании в некоторых ситуациях выражения $(\alpha|\beta)$ вместо $(\alpha \cup \beta)$.

Например, для построения регулярного выражения, задающего множество идентификаторов, можно применить следующую схему

$$\begin{aligned} Letter &= a|b|c|d| \dots |x|y|z \\ Digit &= 0|1|2| \dots |8|9 \\ Identifier &= Letter(Letter|Digit)^*. \end{aligned}$$

Заметим, что хотя с помощью регулярных выражений можно задать достаточно нетривиальные языки, однако нетрудно привести примеры нерегулярных языков, которые весьма просто описываются другими способами. Например, позже будет доказано, что язык $\{0^n 1^n \mid n \geq 0\}$ нельзя задать регулярным выражением. Заметим, что ”близкий” язык $\{0^n 1^m \mid n, m \geq 0\}$ задается регулярным выражением $(0^* \cup 1^*)$.

Конечно, регулярные выражения не могут рассматриваться в качестве достаточно универсального метода задания языков.

Рассмотрим другой подход к заданию языков. При этом мы будем руководствоваться общей схемой задания языков

$$L = \{ w \mid w \in \Sigma^* \text{ \& } w \text{ обладает свойством } P \}.$$

Но ограничимся лишь *разрешимыми* свойствами, то есть такими, для которых существует алгоритм, позволяющий по произвольному слову w определить, выполнено ли для него свойство P .

§3. Конечные автоматы

В этом параграфе речь пойдет о другом подходе к заданию языков – задание языков конечными автоматами. Заметим, что конечные автоматы – одна из трех наиболее распространенных в настоящее время моделей вычислений, используемых для описания и изучения взаимодействия компьютерной программы и компьютера. Две другие – это логические схемы и машины с бесконечной памятью.

Логическая или комбинационная схема – это соединение функциональных элементов, каждый из которых реализует некоторую булеву функцию.

Традиционно считается, что начало изучения комбинационной сложности булевых функций восходит к диссертации Шеннона 1938 года, в которой он показал важность булевой алгебры для синтеза и анализа релейных переключательных схем. Логические схемы способны вычислять функции, однако они не обладают памятью. Конечные автоматы с выходом так же, как и логические схемы, вычисляют функции. Но в отличие от логических схем, конечные автоматы обладают памятью (конечной), что позволяет им использовать свою схемную часть многократно. Последнее дает возможность реализовывать функции с использованием логических схем меньшего размера, чем требуется машинам без памяти.

Конечный автомат – это математическая модель вычислительных машин с конечной памятью, устройство которой известно. У него имеется некоторый аналог процессора реального компьютера – “процессор” фиксированной конечной мощности. Входными данными для конечного автомата служат слова в некотором фиксированном алфавите, называемом его входным или внешним алфавитом. Слова подаются (записываются) на ленту автомата. Различают автоматы с выходом и без выхода. Первые называются автоматами Мили, а вторые – автоматами Мура. Мы будем использовать автоматы лишь в качестве **распознавателей** языков. Поэтому ограничимся автоматами без выхода. Автоматы с выходом нам потребуются при описании теоретико-автоматных моделей шифраторов.

Возникает естественный вопрос “Могут ли конечные автоматы рассматриваться в качестве достаточно адекватной модели реального компьютера?” У конечного автомата имеется память, но ее объем фиксирован “в момент создания” и в дальнейшем не может быть увеличен. А что можно сказать об объеме

памяти реального компьютера? Какая математическая модель лучше описывает – с конечной памятью или с неограниченной? Ответ во многом будет зависеть от целей, с которыми строится модель. Если мы стремимся доказать ограниченность наших алгоритмических возможностей, то, вероятно, целесообразнее использовать модели с неограниченной памятью. Для демонстрации же мощи наших алгоритмических возможностей больше подходят модели с ограниченной памятью. Можно поставить, но не решить еще более общие вопросы – о физических ограничениях, об ограниченности Вселенной.

Теория конечных автоматов достаточно содержательна и находит многочисленные применения. Например, этап *лексического анализа* в компиляторе, когда выделяются лексические единицы программы такие, как служебные слова, идентификаторы, знаки операций и т.д., нередко базируется на построении соответствующих конечных автоматов. Нахождение вхождений одного слова в другое, что часто требуется при редактировании текстов, может быть эффективно выполнено алгоритмом, основанном на теории конечных автоматов.

У конечного автомата имеется *входная лента*, разбитая на ячейки, в каждой из которых можно записывать по одному символу из входного алфавита. Слова во входном алфавите автомата записываются побуквенно в ячейки ленты. Основная часть конечного автомата – *считывающая головка*. Она представляет собой нечто вроде черного ящика и в каждый конкретный момент времени может находиться в одном из конечного числа различных *внутренних состояний* автомата. Считывающая головка может распознать, какой символ записан в обозреваемой ячейке входной ленты. Первоначально считывающая головка размещается на крайнем левом краю ленты и приводится в специальное выделенное *начальное состояние*. Работа автомата осуществляется по тактам. На очередном такте работы автомата считывающая головка читает символ из обозреваемой ячейки входной ленты, переходит в новое состояние, зависящее лишь от текущего состояния и только что прочитанного символа, и сдвигается в соседнюю справа ячейку. Так как новое состояние автомата однозначно определено, то он называется *детерминированным* конечным автоматом. Позже будут введены и *недетерминированные* конечные автоматы. Процесс повторяется, пока не будет прочитано все слово.

С помощью состояния своей головки автомат сообщает, принимается ли прочитанное слово. Для этого выделяется некоторое множество *допускающих состояний*. Если после прочтения слова считывающая головка пришла в одно из допускающих состояний, то входное слово считается *принятым, или распознанным*. *Язык, распознаваемый автоматом*, – это множество всех принимаемых им слов.

Переходим к формальному определению детерминированного конечного автомата.

Определение 3.1. *Детерминированный конечный автомат – это пятерка*

$$M = \langle \Sigma_M, Q_M, \delta_M, s_M, F_M \rangle,$$

где

Σ_M – *входной (внешний) алфавит* автомата M ,

Q_M – *внутренний алфавит* автомата M , его элементы называются *внутренними состояниями*,

$s_M \in Q_M$ – *начальное состояние* автомата M ,

$F_M \subseteq Q_M$ – *множество конечных (заключительных, допускающих) состояний* автомата M ,

δ_M – *функция перехода* автомата M , это функция из $Q_M \times \Sigma_M$ в Q_M .

В ячейки ленты автомата M записываются последовательно буквы внешнего алфавита Σ_M , таким образом на ленте оказывается записанным некоторое слово w во внешнем алфавите Σ_M автомата M . Конечный автомат M работает (функционирует) в дискретном режиме времени, потактно. В начале каждого такта считывающая головка находится в некотором внутреннем состоянии $q \in Q_M$ и обозревает некоторую ячейку ленты. Если в этой ячейке находится (прочитывается) символ $a \in \Sigma_M$, то считывающая головка (автомат) переходит в состояние $\delta_M(q, a) \in Q_M$ и сдвигается в соседнюю справа ячейку ленты.

Вычисление, производимое автоматом при данном входе, – это последовательность **конфигураций**, которые отражают состояние автомата в последовательные моменты времени. Детерминированный конечный автомат не может сдвигать свою головку назад на уже прочитанную часть входного слова. Поэтому часть слова слева от считывающей головки не влияет на дальнейшее функционирование автомата. Значит, *конфигурация автомата в каждый момент времени определяется текущим состоянием и непрочитанной частью обрабатываемого слова*. Это оправдывает следующее определение.

Определение 3.2. *Конфигурация детерминированного конечного автомата M – это любое слово вида qw , где $q \in Q_M$, а $w \in \Sigma^*$.*

Замечание. Иногда мы будем конфигурацию представлять в виде $\langle q, w \rangle$, т.е. как элемент множества $Q_M \times \Sigma^*$.

На множестве всех конфигураций введем бинарное отношение \vdash_M , которое будет выполняться для двух конфигураций тогда и только тогда, когда **автомат может перейти из одной из них в другую за один шаг**. Более точно, если qw и $q'w'$ – две конфигурации автомата M , то

$qw \vdash_M q'w'$ тогда и только тогда, когда

$$w = aw' \text{ для некоторого символа } a \in \Sigma, \text{ и } \delta(q, a) = q'.$$

В этом случае будем говорить, что **автомат переходит за один шаг от конфигурации (из конфигурации) qw к конфигурации (в конфигурацию) $q'w'$** . Так как для каждой конфигурации (кроме заключительной, имеющей форму $\langle q, \varepsilon \rangle$) существует единственная последующая конфигурация, то фактически \vdash_M – это функция из $Q_M \times \Sigma^+$ в $Q_M \times \Sigma^*$.

Обозначим через \vdash_M^* *рефлексивное, транзитивное замыкание* отношения \vdash_M , т.е. $qw \vdash_M^* q'w'$ тогда и только тогда, когда от конфигурации qw можно перейти к конфигурации $q'w'$ за конечное число шагов. В частности выполнено $qw \vdash_M^* qw$, что означает возможность перейти от конфигурации qw к конфигурации qw за нуль шагов.

Определение 3.3. Слово $w \in \Sigma^*$ *принимается (допускается, распознается)* автоматом M тогда и только тогда, когда существует такое состояние $q \in F_M$, что $s_M w \vdash_M^* q\varepsilon$.

Язык $L(M)$, принимаемый (допускаемый, распознаваемый) автоматом M , – это множество всех слов, им распознаваемых.

Функцию δ_M переходов автомата M , определенную на $Q_M \times \Sigma$, можно продолжить до функции δ_M^* , определенной на $Q_M \times \Sigma^*$, полагая

$$\delta_M^*(q, \varepsilon) = q, \quad \delta_M^*(q, aw) = \delta_M^*(\delta_M(q, a), w), \text{ где } a \in \Sigma, w \in \Sigma^*.$$

Тогда язык $L(M)$, принимаемый автоматом M , задается равенством

$$L(M) = \{ w \mid w \in \Sigma^* \ \& \ \delta_M^*(s_M, w) \in F_M \}.$$

Функцию δ_M переходов автомата M можно задать таблицей, строки которой помечены внутренними состояниями автомата, а столбцы – символами его внешнего алфавита. На пересечении строки, помеченной состоянием q , и столбца, помеченного символом a , стоит состояние $\delta_M(q, a)$.

Однако для задания автомата удобнее использовать более наглядное графическое представление, именуемое **диаграммой состояний** или **диаграммой переходов**. Это размеченный ориентированный граф, вершины которого соответствуют состояниям автомата и ими помечены, а дуги графа соответствуют переходам между состояниями. Более точно, из вершины, помеченной состоянием q , берет начало дуга, помеченная символом a и заканчивающаяся в вершине $\delta_M(q, a)$. Конечные состояния изображаются двойными кружками, а начальное – значком $>$.

Для облегчения изучения автоматных языков рассмотрим некоторое обобщение понятия автомата – понятие **недетерминированного** автомата. Детерминированные автоматы реализуют те или иные алгоритмы, представляющие собой *”наборы приказов на исполнение”*. Недетерминированные автоматы будут включать в себя *”наборы разрешенных действий”*, что роднит их с более общим понятием *исчисления*. Заметим, что с идеей недетерминированности при выполнении алгоритмов связан ряд нерешенных на сегодняшний день проблем, самая знаменитая из которых – это проблема равенства классов **NP** и **P**, открывающая список *”Проблемы тысячелетия”*. **Недетерминированность** будет состоять в способности автомата переходить в одно из нескольких возможных состояний, т.е. новое состояние частично зависит от текущего

состояния и читаемого символа – допускается наличие в программе автомата для данной комбинации текущего состояния и входного символа нескольких возможных следующих состояний. Поэтому автомат, читая побуквенно входное слово, может на каждом шаге ”выбирать”, в какое из разрешенных состояний перейти. Этот выбор ничем не предопределен, поэтому называется **недетерминированным**. Однако выбор может осуществляться только среди тех следующих состояний, в которые возможно перейти из данного состояния при данном входном символе.

Насколько подобные недетерминированные устройства физически реализуемы, сегодня трудно сказать. Возможно, это будет выяснено в ходе интенсивно ведущихся в настоящее время исследований, связанных с квантовыми компьютерами, для которых идея недетерминированности является базовой. Но можно утверждать, что недетерминированные автоматы представляют собой весьма полезные обобщения конечных автоматов, так как в значительной степени упрощают изучение детерминированных автоматов. Кроме того, будет показано, что в случае конечных автоматов недетерминированность не является существенной: *каждый недетерминированный конечный автомат эквивалентен детерминированному конечному автомату*.

Определение 3.4. *Недетерминированный конечный автомат – это пятерка*

$$M = \langle \Sigma_M, Q_M, \Delta_M, s_m, F \rangle,$$

где

Σ_M – *входной (внешний) алфавит* автомата M ,

Q_M – *внутренний алфавит* автомата M , его элементы называются **внутренними состояниями**,

$s_m \in Q_M$ – *начальное состояние* автомата M ,

$F_M \subseteq Q_M$ – *множество конечных (заключительных, допускающих) состояний* автомата M ,

Δ_M – *отношение переходов* автомата M , это подмножество множества $Q_M \times (\Sigma \cup \{\varepsilon\}) \times Q_M$.

Любая тройка $\langle q, u, p \rangle$ из Δ_M называется **возможным переходом** для автомата M . Если в текущий момент времени автомат M находится в состоянии q , считывающая головка обзрывает символ a , то автомат M может выполнить любой переход вида $\langle q, a, p \rangle$ или $\langle q, \varepsilon, p \rangle$ при условии, что $\langle q, a, p \rangle \in \Delta_M$, $\langle q, \varepsilon, p \rangle \in \Delta_M$. Если выполняется переход вида $\langle q, \varepsilon, p \rangle$, то считывающая головка остается на месте, т.е. фактически считывания символа не происходит.

Определение вычисления, выполняемого недетерминированным конечным автоматом, незначительно отличается от соответствующего определения для детерминированных конечных автоматов. **Конфигурация** автомата M – это, как и ранее, слово вида qw , где $q \in Q_M$, а $w \in \Sigma^*$ или элемент $\langle q, w \rangle$ из $Q_M \times \Sigma^*$. Отношение \vdash_M между конфигурациями ”**переход за один шаг**” определяется

аналогично детерминированному случаю:

$qw \vdash_M ri$ тогда и только тогда, когда
существует $v \in \Sigma_M \cup \{\varepsilon\}$ такой, что $w = vi$ и $\langle q, v, p \rangle \in \Delta_M$.

В отличие от детерминированного случая для недетерминированного автомата M отношение \vdash_M может не быть функцией: для некоторых конфигураций qw может быть несколько пар ri или не быть ни одной такой пары, что $qw \vdash_M ri$.

Как и в детерминированном случае, через \vdash_M^* обозначим рефлексивное и транзитивное замыкание \vdash_M . Слово $w \in \Sigma_M^*$ **принимается (распознается, допускается)** автоматом M тогда и только тогда, когда существует состояние $p \in F_M$ такое, что $s_M w \vdash_M^* p\varepsilon$. Язык $L(M)$, **принимаемый (расознаваемый, допускаемый) автоматом M** , – это множество слов, распознаваемых автоматом M .

Как и в детерминированном случае, отношение Δ_M переходов автомата M можно задать таблицей, строки которой помечены внутренними состояниями автомата, а столбцы – символами его внешнего алфавита. На пересечении строки, помеченной состоянием q , и столбца, помеченного символом v из $\Sigma \cup \{\varepsilon\}$, стоят все те состояния (если таковые имеются), в которые возможен переход из конфигурации qv .

Однако и в этом случае для задания автомата удобнее использовать более наглядное графическое представление – **диаграмму состояний** или **диаграмму переходов**. Как и в детерминированном случае, это размеченный ориентированный граф, вершины которого соответствуют состояниям автомата и ими помечены, а дуги графа соответствуют возможным переходам между состояниями. Более точно, *из вершины, помеченной состоянием q , берет начало дуга, помеченная символом v из $\Sigma \cup \{\varepsilon\}$ и заканчивающаяся в вершине p тогда и только тогда, когда $\langle q, v, p \rangle \in \Delta_M$* . Конечные состояния изображаются двойными кружками, а начальное – значком " $>$ ".

Детерминированный конечный автомат можно рассматривать как специальную разновидность недетерминированного автомата, у которого отношение переходов Δ_M является функцией из $Q_M \times \Sigma$ в Q_M . Недетерминированный конечный автомат $\langle \Sigma_M, Q_M, \Delta_M, s_M, F_M \rangle$ является детерминированным тогда и только тогда, когда его отношение переходов Δ_M не содержит троек вида $\langle q, \varepsilon, p \rangle$ с $p \neq q$, и для любого $q \in Q_M$ и $a \in \Sigma_M$ существует единственное $p \in Q_M$ такое, что $\langle q, a, p \rangle \in \Delta_M$. Значит, класс языков, распознаваемых детерминированными автоматами, является подклассом класса языков, распознаваемых недетерминированными автоматами. На самом деле, эти два класса совпадают. С точки зрения возможности распознавания языков недетерминированные автоматы не являются более мощным аппаратом, чем детерминированные, – каждый недетерминированный конечный автомат может быть преобразован в эквивалентный ему детерминированный. Однако недетерминированные автоматы существенно облегчают задачу построения детерминированных автоматов, распознающих тот или иной язык: нередко бывает

достаточно сложно непосредственно построить детерминированный конечный автомат, распознающий некоторый важный с той или иной точки зрения язык. Это построение можно выполнить в два этапа: сначала построить распознающий этот язык недетерминированный конечный автомат, что часто бывает сделать значительно проще, а затем его преобразовать в детерминированный.

Определение 3.5. Конечные автоматы M_1 и M_2 называются **эквивалентными**, если распознаваемые ими языки $L(M_1)$ и $L(M_2)$ совпадают.

Теорема 3.1. Для любого недетерминированного конечного автомата существует эквивалентный ему детерминированный конечный автомат.

Д о к а з а т е л ь с т в о. Пусть $M = \langle \Sigma_M, Q_M, \Delta_M, s_M, F_M \rangle$ – недетерминированный конечный автомат. Построим детерминированный конечный автомат $M' = \langle \Sigma_{M'}, Q_{M'}, \delta_{M'}, s_{M'}, F_{M'} \rangle$, эквивалентный M . Идея построения такого автомата заключается в том, чтобы рассматривать недетерминированный конечный автомат в данный момент времени не как находящийся в некотором определенном состоянии, а как находящийся одновременно во множестве состояний – во всех тех состояниях, которые могут быть достигнуты от начального при прочитанной к текущему моменту части входного слова.

Внешний (входной) алфавит автомата M' совпадает с внешним алфавитом автомата M , т.е. $\Sigma_{M'} = \Sigma_M$. Множеством внутренних состояний автомата M' является множество $Q_{M'} = P(Q_M)$ всех подмножеств множества Q_M внутренних состояний автомата M . Множество $F_{M'}$ заключительных состояний автомата M' определяется как множество, состоящее из всех тех подмножеств множества Q_M , которые содержат хотя бы одно заключительное состояние M , т.е.

$$F_{M'} = \{U \mid U \subseteq Q_M \text{ и } U \cap F_M \neq \emptyset\}.$$

Для определения функции переходов $\delta_{M'}$ для автомата M' потребуется одно вспомогательное понятие. Для любого состояния $q \in Q_M$ недетерминированного автомата M через $E(q)$ обозначается множество всех состояний этого автомата M , которые можно достичь из состояния q , читая лишь пустое слово (без чтения ввода), т.е.

$$E(q) = \{p \mid p \in Q_M \text{ и } q \varepsilon \vdash_M^* p \varepsilon\}.$$

Легко указать алгоритм для построения множества $E(q)$.

Определим начальное состояние автомата M' равенством $s_{M'} = E(s)$. Для произвольного состояния $Q \in Q_{M'}$ автомата M' , т.е. подмножества $Q \subseteq Q_M$ множества состояний автомата M , и каждого символа $a \in \Sigma_{M'}$ его внешнего алфавита полагаем

$$\delta_{M'}(Q, a) = \bigcup_{\substack{q \in Q \\ \langle q, a, p \rangle \in \Delta_M}} E(p).$$

Содержательно $\delta_{M'}(Q, a)$ – это множество всех тех состояний автомата M , которые он может достичь, прочитав символ a (и сделав несколько переходов по пустому слову).

Покажем, что детерминированный автомат M' эквивалентен недетерминированному автомату M .

Основу доказательства составляет следующее утверждение:

для произвольного слова w в алфавите $\Sigma_{M'}$ автоматов M и M' и любых двух состояний p и q автомата M справедлива эквивалентность:

$$qw \vdash_M^* p\varepsilon \quad \text{тогда и только тогда, когда} \quad (\exists P \in Q_{M'}) E(q)w \vdash_{M'}^* P\varepsilon \ \& \ p \in P.$$

Так как P , о котором идет речь в этой эквивалентности, удовлетворяет равенству $P = \delta_{M'}^*(E(q), w)$, то доказываемая эквивалентность принимает вид

$$qw \vdash_M^* p\varepsilon \quad \text{тогда и только тогда, когда} \quad p \in \delta_{M'}^*(E(q), w).$$

Доказательство этого утверждения проведем индукцией по длине слова w .

База индукции. Если $|w| = 0$, то $w = \varepsilon$ – пустое слово. Поэтому доказываемая эквивалентность следует из следующих эквивалентности и равенства

$$q\varepsilon \vdash_M^* p\varepsilon \quad \text{тогда и только тогда, когда} \quad p \in E(q), \quad \delta_{M'}^*(E(q), \varepsilon) = E(q).$$

Индуктивный переход. Пусть слово w имеет вид va , где $v \in \Sigma_{M'}^*$ и $a \in \Sigma_{M'}$.

Допустим, что $qw = qva \vdash_M^* p\varepsilon$. Тогда найдутся два таких состояния p_1 и p_2 автомата M , что

$$qw = qva \vdash_M^* p_1a \vdash_M p_2\varepsilon \vdash_M^* p\varepsilon.$$

Переход $p_1a \vdash_M p_2\varepsilon$ – это момент чтения буквы a .

Так как

$$qva \vdash_M^* p_1a \iff qv \vdash_M^* p_1\varepsilon,$$

то по индуктивному предположению

$$E(q)v \vdash_{M'}^* P_1\varepsilon \ \& \ p_1 \in P_1 = \delta_{M'}^*(E(q), v).$$

Кроме того

$$p_1a \vdash_M p_2\varepsilon \iff \langle p_1, a, p_2 \rangle \in \Delta_{M'}.$$

Поэтому

$$\delta_{M'}(P_1, a) = \bigcup_{\substack{t \in P_1 \\ \langle t, a, t' \rangle \in \Delta_M}} \supseteq E(p_2).$$

Так как

$$p_2\varepsilon \vdash_M^* p\varepsilon \iff p \in E(p_2),$$

то

$$p \in \delta_{M'}(P_1, a) = \delta_{M'}(\delta_{M'}^*(E(q), v), a) = \delta_{M'}^*(E(q), va) = \delta_{M'}^*(E(q), w),$$

т.е. $p \in \delta_{M'}^*(E(q), w)$.

Для доказательства обратного утверждения предположим, что

$$p \in \delta_{M'}^*(E(q), va).$$

Полагаем

$$P = \delta_{M'}^*(E(q), va), P_1 = \delta_{M'}^*(E(q), v),$$

тогда $p \in P = \delta_{M'}^*(P_1, a)$ и выполнены переходы

$$E(q)va \vdash_{M'}^* P_1a \vdash_{M'} P\varepsilon.$$

Так как

$$P = \delta_{M'}(P_1, a) = \bigcup_{\substack{p_1 \in P_1 \\ \langle p_1, a, p_2 \rangle \in \Delta_M}} E(p_2),$$

то найдутся такие состояния p_1 и p_2 автомата M , что $p_1 \in P_1$, $\langle p_1, a, p_2 \rangle \in \Delta_M$ и $p \in E(p_2)$. Значит $p_2\varepsilon \vdash_{M'}^* p\varepsilon$. Из $E(q)v \vdash_{M'}^* P_1\varepsilon$ и $p_1 \in P_1$ по индуктивному предположению следует, что $qv \vdash_M^* p_1\varepsilon$. В итоге получаем цепочку переходов

$$qva \vdash_M^* p_1a \vdash_M p_2\varepsilon \vdash_M^* p\varepsilon.$$

Значит $qva \vdash_M^* p\varepsilon$.

Докажем эквивалентность автоматов M и M' , т.е. совпадение языков $L(M)$ и $L(M')$.

Если w – слово во внешнем алфавите этих автоматов, то $w \in L(M)$ тогда и только тогда, когда найдется $f \in F_M$ такое, что $s_M w \vdash_M^* f\varepsilon$. В силу уже доказанного последнее равносильно $f \in \delta_{M'}^*(E(s_M), w)$.

В итоге получаем эквивалентность $w \in L(M)$ тогда и только тогда, когда $\delta_{M'}^*(E(s_M), w) \cap F_M \neq \emptyset$. А последнее эквивалентно $\delta_{M'}^*(s_{M'}, w) \in F_{M'}$, т.е. $w \in L(M')$, поэтому $L(M) = L(M')$. \square

Замечание. Доказательство предыдущей теоремы является фактически **конструктивным** – из него можно извлечь **алгоритм** построения по любому недетерминированному конечному автомату M эквивалентного ему детерминированного автомата M' .

§4. Конечные автоматы и регулярные выражения

В этом параграфе будет установлено, что рассмотренные выше два варианта конечных заданий языков путем их описания регулярными выражениями и заданием реализуемых конечными автоматами алгоритмов, распознающих эти

языки, на самом деле эквивалентны. Это утверждение носит название теоремы С. Клини. При этом регулярные выражения выступают в качестве описаний (описателей) языков, а конечные автоматы – в качестве их распознавателей. Эти два подхода к заданию языков в определенном смысле дополняют друг друга. Чтобы излишне не удлинять доказательство теоремы С. Клини, предварительно докажем утверждение, которое составит основную часть этого доказательства и важное само по себе.

Теорема 4.1. *Класс языков в алфавите Σ , распознаваемых конечными автоматами, замкнут относительно теоретико-множественных операций*

- (1) объединение,
- (2) пересечение,
- (3) разность,
- (4) конкатенация;
- (5) навешивание звездочки Клини.

До к а з а т е л ь с т в о. Доказательство теоремы проведем по следующей схеме:

в каждом из первых четырех случаев покажем, как по двум конечным автоматам M_1 и M_2 , распознающим языки L_1 и L_2 , построить конечный автомат M , распознающий язык L , получающийся из L_1 и L_2 с помощью рассматриваемой операции; в пятом случае – навешивание звездочки Клини – покажем, как по конечному автомату M , распознающему язык L , построить конечный автомат M^* , распознающий язык L^* , получающийся из языка L навешиванием звездочки Клини.

(1) **Объединение.**

По произвольным двум недетерминированным конечным автоматам

$$M_1 = \langle \Sigma, Q_{M_1}, \Delta_{M_1}, s_{M_1}, F_{M_1} \rangle,$$

$$M_2 = \langle \Sigma, Q_{M_2}, \Delta_{M_2}, s_{M_2}, F_{M_2} \rangle$$

построим недетерминированный конечный автомат M , распознающий язык $L(M_1) \cup L(M_2)$. Идея построения автомата M проста: используя недетерминизм, автомат M , получив входное слово w , решает, "какую из двух гипотез проверять": " $w \in L(M_1)$?" или " $w \in L(M_2)$?", т.е. "определяет, какому из двух автоматов M_1 и M_2 передать его на обработку", затем обрабатывает слово так, как это бы сделал выбранный им автомат.

Построим автомат M , реализующий этот проект. Считаем, что множества Q_{M_1} и Q_{M_2} внутренних состояний автоматов M_1 и M_2 не пересекаются. Конечный недетерминированный автомат M , распознающий язык $L(M_1) \cup L(M_2)$, можно задать следующим образом:

$$M = \langle \Sigma, Q_M, \Delta_M, s_M, F_M \rangle,$$

где s_M – новое состояние, не входящее в $Q_{M_1} \cup Q_{M_2}$,

$$\begin{aligned} Q_M &= Q_{M_1} \cup Q_{M_2} \cup \{s_M\}, \\ F_M &= F_{M_1} \cup F_{M_2}, \\ \Delta_M &= \Delta_{M_1} \cup \Delta_{M_2} \cup \{\langle s_M, \varepsilon, s_{M_1} \rangle, \langle s_M, \varepsilon, s_{M_2} \rangle\}. \end{aligned}$$

Автомат M , получив в качестве входа слово w , начинает работу с недетерминированного выбора: перейти в начальное состояние s_{M_1} автомата M_1 или в начальное состояние s_{M_2} автомата M_2 . Затем он работает как M_1 или M_2 . Значит, $s_M w \vdash_M^* p\varepsilon$ для некоторого p из F_M тогда и только тогда, когда или $s_{M_1} w \vdash_{M_1}^* p\varepsilon$ для некоторого p из F_{M_1} или $s_{M_2} w \vdash_{M_2}^* p\varepsilon$ для некоторого p из F_{M_2} . Таким образом, автомат M принимает (распознает) слово w тогда и только тогда, когда это слово принимает автомат M_1 или автомат M_2 . Значит, $L(M) = L(M_1) \cup L(M_2)$.

(2) Пересечение.

Для доказательства замкнутости класса автоматных языков относительно операций пересечения и разности воспользуемся следующей схемой: докажем замкнутость этого класса относительно операции взятия дополнения $\bar{L} = \Sigma^* \setminus L$ и воспользуемся хорошо известными равенствами

$$L_1 \setminus L_2 = L_1 \cap \bar{L}_2, \quad L_1 \cap L_2 = \overline{\bar{L}_1 \cup \bar{L}_2}.$$

Пусть **детерминированный** конечный автомат $M = \langle \Sigma, Q_M, \delta_M, s_M, F_M \rangle$ распознает язык L . Тогда дополнение этого языка $\bar{L} = \Sigma^* \setminus L(M)$ распознается детерминированным конечным автоматом

$$\bar{M} = \langle \Sigma, Q_M, \Delta_M, s_M, Q_M \setminus F_M \rangle.$$

(4) Конкатенация.

По недетерминированным конечным автоматам M_1 и M_2 построим недетерминированный конечный автомат M такой, что $L(M) = L(M_1) \circ L(M_2)$. Идея конструкции этого автомата проста: вначале автомат M работает как автомат M_1 (имитирует работу M_1), достигнув одного из заключительных состояний автомата M_1 , автомат M может продолжить работу как автомат M_1 , а может недетерминированно "перескочить" в начальное состояние автомата M_2 , после этого он работает как автомат M_2 (имитирует работу M_2).

Построим автомат M , реализующий этот проект. Считаем, что множества Q_{M_1} и Q_{M_2} внутренних состояний автоматов M_1 и M_2 не пересекаются. Конечный недетерминированный автомат M , распознающий язык $L(M_1) \cdot L(M_2)$, можно задать следующим образом:

$$M = \langle \Sigma, Q_M, \Delta_M, s_M, F_M \rangle,$$

где $s_M = s_{M_1}$, $Q_M = Q_{M_1} \cup Q_{M_2}$, $F_M = F_{M_2}$,

$$\Delta_M = \Delta_{M_1} \cup \Delta_{M_2} \cup \{\langle q, \varepsilon, s_{M_2} \rangle \mid q \in F_{M_1}\}.$$

(5) **Навешивание звездочки Клини.**

По недетерминированному конечному автомату M_1 , распознающему язык $L(M_1)$, построим недетерминированный конечный автомат M такой, что $L(M) = L(M_1)^*$. Автомат M включает в себя все состояния автомата M_1 и все его переходы. Все конечные состояния автомата M_1 являются и конечными состояниями автомата M . Дополнительно у M есть новое начальное состояние — s_M . Оно является и одним из конечных, чтобы допустить пустое слово. Из начального состояния s_M имеется ε -переход в начальное состояние s_{M_1} автомата M_1 . Начав работать в своем начальном состоянии s_M , автомат M может либо допустить пустое слово, либо начать имитировать работу автомата M_1 . Из каждого конечного состояния автомата M_1 добавляется ε -переход обратно в начальное состояние s_{M_1} автомата M_1 . Поэтому, приняв некоторое начало входного слова w как принадлежащее языку $L(M_1)$, автомат может перейти в начальное состояние s_{M_1} автомата M_1 и продолжить процесс распознавания оставшейся непрочитанной части слова.

Построим автомат M , реализующий этот проект.

$$M = \langle \Sigma, Q_M, \Delta_M, s_M, F_M \rangle,$$

где s_M — новое состояние, не входящее в Q_{M_1} , $Q_M = Q_{M_1} \cup \{s_M\}$, $F_M = F_{M_1} \cup \{s_M\}$,

$$\Delta_M = \Delta_{M_1} \cup \{ \langle s_M, \varepsilon, s_{M_1} \rangle \cup \{ \langle q, \varepsilon, s_{M_1} \rangle \mid q \in F_{M_1} \}.$$

□

Докажем одну из основных теорем теории конечных автоматов.

Теорема 4.2 (С. Клини). *Язык является регулярным тогда и только тогда, когда он распознается некоторым конечным автоматом.*

До к а з а т е л ь с т в о. Так как класс автоматных языков замкнут относительно объединения, конкатенации и звездочки Клини, то для доказательства автоматности любого регулярного языка в соответствии с определением класса регулярных языков достаточно установить автоматность пустого языка, языка, состоящего лишь из пустого слова и однобуквенных языков. Построение соответствующих автоматов тривиально. Поэтому можно считать доказанным, что *каждый регулярный язык распознается некоторым конечным автоматом.*

Для доказательства обратного докажем, что по любому конечному автомату $M = \langle \Sigma, Q_M, \Delta_M, s_M, F_M \rangle$ можно построить такое регулярное выражение α , что $L(M) = L(\alpha)$.

Представим $L(M)$ в виде объединения конечного числа более простых языков.

Пусть $Q_M = \{q_1, \dots, q_n\}$ и $s_M = q_1$. Для произвольных чисел i, j и k , где $i, j = 1, \dots, n$ и $k = 0, \dots, n$ определим язык $L(i, j, k)$ как множество всех слов из Σ^* , которые могут перевести автомат M из состояния q_i в состояние q_j без

прохождения через промежуточное состояние с номером $k + 1$ или больше при этом конечные точки q_i и q_j могут иметь любые номера. При $k = n$ получим

$$L(i, j, n) = \{ w \mid w \in \Sigma^* \& q_i w \vdash_M^* q_j \varepsilon \}.$$

Так как

$$L(M) = \bigcup_{q_j \in F_M} L(1, j, n),$$

то если будет установлена регулярность всех языков $L(i, j, k)$, регулярным будет и язык $L(M)$.

Регулярность языков $L(i, j, k)$ доказывается индукцией по k .

Если $k = 0$, то возможны два случая. При $i \neq j$ $L(i, j, 0) = \{ a \mid a \in \Sigma \cup \{ \varepsilon \} \& \langle q_i, a, q_j \rangle \in \Delta_M \}$. Если же $i = j$, то $L(i, j, 0) = \{ \varepsilon \} \cup \{ \sigma \mid \sigma \in \Sigma_M \cup \{ \varepsilon \} \& \langle q_i, \sigma, q_j \rangle \in \Delta_M \}$. Каждое из этих множеств, будучи конечным, является регулярным.

Если для всех i, j уже установлена регулярность языков $L(i, j, k - 1)$, то регулярность языка $L(i, j, k)$ следует из равенства

$$L(i, j, k) = L(i, j, k - 1) \cup L(i, k, k - 1) \cdot L(k, k, k - 1)^* \cdot L(k, j, k - 1).$$

Это равенство просто означает, что при движении из q_i в q_j без прохождения через состояния с номерами, большими чем k , у автомата M есть две возможности:

- (1) идти из q_i в q_j без прохождения через состояния с номерами, большими чем $k - 1$,
- (2) идти из q_i в q_k , затем из q_k в q_k 0 или более раз, затем идти из q_k в q_j , при этом каждый раз без прохождения через промежуточные состояния с номерами, большими чем $k - 1$.

Значит, при любых i, j, k язык $L(i, j, k)$ регулярен. Поэтому регулярным является и язык $L(M)$. Это завершает доказательство теоремы. \square

Замечание. Из доказательства теоремы можно извлечь алгоритм построения по конечному автомату, задающему некоторый язык, регулярного выражения, описывающего тот же язык.

Имеется немало средств для установления регулярности языков – это и регулярные выражения, и конечные детерминированные и недетерминированные автоматы.

Из теоретико-множественных соображений следует, что *нерегулярные языки существуют*, так как множество регулярных выражений над фиксированным конечным или счетным алфавитом счетно, а множество всех языков несчетно. Но для доказательства нерегулярности конкретного языка требуются дополнительные знания свойств регулярных языков. Ситуация отдаленно напоминает возникшую в конце XIX века ситуацию в области трансцендентных чисел: из теоремы Г. Кантора о несчетности множества всех действительных чисел и счетности множества алгебраических чисел следует несчетность множества

всех трансцендентных чисел, однако установление трансцендентности конкретных чисел, как правило, было связано с решением крупной математической проблемы. Напомним лишь о доказательстве Ш. Эрмитом трансцендентности числа e и о выдающемся успехе Ф. Линдемана, доказавшего трансцендентность числа π .

Следующая важная теорема позволяет устанавливать нерегулярность некоторых конкретных языков, а также полезна при решении ряда алгоритмических проблем, связанных с регулярными языками.

Теорема 4.3 (Теорема о накачке (о насосе)). *Для любого регулярного языка L существует такое число n , что каждое слово w из L длины $|w| \geq n$ может быть представлено в виде $w = xyz$, причем $y \neq \varepsilon$, $|xy| \leq n$ и для каждого $k \geq 0$ $xy^kz \in L$.*

До к а з а т е л ь с т в о. Будучи регулярным, язык L распознается некоторым детерминированным конечным автоматом M . Пусть n – число состояний этого автомата M , а q_0 – его начальное состояние.

Пусть слово w длины $|w| = m \geq n$ принадлежит языку L . Рассмотрим работу автомата M на входном слове w :

$$q_0 w_1 w_2 \dots w_m \vdash_M q_1 w_2 \dots w_m \vdash_M \dots \vdash_M q_m \varepsilon,$$

где w_1, \dots, w_m – это буквы слова w . Заметим, что $q_m \in F_M$.

Так как у автомата M лишь n состояний, а в указанном вычислении возникает $m + 1$ конфигурация вида $q_i w_{i+1} \dots w_m$ и $m + 1 > n$, то существуют такие числа i и j , что $0 \leq i < j \leq n$ и $q_i = q_j$. Поэтому слово $y = w_{i+1} \dots w_j$ переводит автомат M из состояния q_i назад в это же состояние q_i . Так как $i < j$, то это слово непусто. Введем обозначения $x = w_1 \dots w_i$ и $z = w_{j+1} \dots w_m$. Ясно, что слово $xz = w_1 \dots w_i w_{j+1} \dots w_m$ переводит автомат M из состояния q_0 в то же самое принимающее состояние q_m , что и слово $w = xyz$. Значит $xz \in L$.

При любом $k \geq 1$ слово

$$xy^kz = w_1 \dots w_i \cdot w_{i+1} \dots w_j \cdot w_{i+1} \dots w_j \dots w_{i+1} \dots w_j \cdot w_{j+1} \dots w_m$$

переводит автомат M из состояния q_0 в то же самое принимающее состояние q_m , что и слово $w = xyz$. Значит $xy^kz \in L$. \square

В качестве примеров применения теоремы о накачке установим нерегулярность некоторых по разным причинам важных и интересных языков.

Пример 1. Язык $L_1 = \{a^k b^k | k \geq 0\}$ не является регулярным. В самом деле, предположим, что он является регулярным. Пусть n – число, о котором идет речь в теореме о накачке. Рассмотрим принадлежащее этому языку слово $w = a^n b^n$. По теореме о накачке его можно представить в виде $w = xyz$ так, что $y \neq \varepsilon$ и при любом $k \geq 0$ $xy^kz \in L$.

Если $y = a^i$ для некоторого $i > 0$, то $xz = a^{n-i} b^n \notin L$, что противоречит теореме о накачке.

Если $y = b^i$ для некоторого $i > 0$, то $xz = a^n b^{n-i} \notin L$, что противоречит теореме о накачке.

Если $y = a^i b^j$ для некоторого i, j , то $xy^k z = a^{n-i} (a^i b^j)^k b^{n-j} \notin L$, что также противоречит теореме о накачке.

Пример 2. Рассмотрим язык

$$L_0 = \{w \mid w \in \{a, b\}^* \text{ \& слово } w \text{ имеет равное количество букв } a \text{ и } b\}.$$

Если бы язык L_0 был регулярным, то таковым бы был и язык $L_0 \cap L(a^* b^*)$. Однако нерегулярность этого языка уже установлена в примере 1.

§5. Некоторые алгоритмические проблемы для языков

Для языков в алфавите Σ естественным образом формулируются алгоритмические проблемы:

Проблема вхождения: по произвольному языку L в алфавите Σ и произвольному слову w в этом алфавите требуется определить, принадлежит ли слово w языку L .

В такой чрезвычайно общей постановке проблема вхождения, конечно, не может быть решена. Необходимо прежде всего уточнить, как задан язык L . Естественным ограничиться рассмотрением лишь языков с "конечным заданием", дающим в определенном смысле основную информацию о языке. Примерами таких конечных заданий могут служить регулярные выражения и конечные автоматы. В этом случае получаем следующую проблему.

Проблема вхождения для конечно-автоматных (регулярных) языков: по произвольному автоматному (регулярному) языку L в алфавите Σ и произвольному слову w в этом алфавите требуется определить, принадлежит ли слово w языку L .

Сделаем некоторые пояснения. Конечно, ввиду совпадения классов конечно-автоматных и регулярных языков можно было бы в формулировке проблемы вхождения ограничиться лишь одним из этих двух понятий. И это оправдано в ситуации, пока нас интересует лишь сам факт существования соответствующего алгоритма. Однако позже нас будет интересовать не только вопрос о существовании алгоритма, но и характеристики его сложности – время работы, объем используемой памяти, сложность задания (описания) алгоритма и т.п. Время работы алгоритма и объем используемой памяти являются функциями от **размера входных данных**. И если **размером** входного слова w естественно считать его длину $|w|$, то **размером** языка L естественно считать **длину** его **задания**. В таком случае выбор способа задания языка L может оказаться существенным.

При фиксировании языка L мы получаем частную проблему вхождения в этот язык.

Частная проблема вхождения для языка L : для фиксированного языка L в алфавите Σ по произвольному слову w в этом алфавите требуется определить, принадлежит ли слово w языку L .

Без доказательства заметим, что можно привести важные примеры языков с различными конечными заданиями, для которых даже частная проблема вхождения алгоритмически неразрешима. Однако подробное обсуждение этого вопроса увело бы нас слишком далеко от основной темы.

Доказательство следующей теоремы очевидно, но мы все же отметим эту теорему.

Теорема 5.1. *Проблема вхождения для класса конечно-автоматных (регулярных) языков алгоритмически разрешима.*

Доказательство. Если конечно-автоматный (регулярный) язык L распознается конечным детерминированным автоматом

$$M = \langle \Sigma_M, Q_M, \delta_M, s_M, F_M \rangle,$$

то для ответа на вопрос, принадлежит ли слово w языку L , достаточно вычислить значение функции $\delta_M^*(s_M, w)$ и проверить, принадлежит ли оно множеству F_M допускающих состояний автомата M . \square

В качестве другой алгоритмической проблемы для языков традиционно рассматривается проблема пустоты языка.

Проблема пустоты языка: по произвольному языку L в алфавите Σ требуется определить, пуст ли язык L .

Теорема 5.2. *Проблема пустоты для класса конечно-автоматных (регулярных) языков алгоритмически разрешима.*

Доказательство. Если конечно-автоматный (регулярный) язык L распознается конечным детерминированным автоматом

$$M = \langle \Sigma_M, Q_M, \delta_M, s_M, F_M \rangle,$$

то для ответа на вопрос, пуст ли этот язык, можно предложить, например, следующие два алгоритма.

Первый алгоритм основан на теореме о накачке, из которой следует, что если язык L , распознаваемый конечным автоматом M с n состояниями, непуст (содержит некоторое слово), то он содержит слово длины, не превосходящей n . Для проверки пустоты языка остается перебрать слова длины, не превосходящей n , и для каждого из них решить проблему вхождения. Если при таком переборе мы обнаружим слово, входящее в язык L , то L – непустой язык, а если не обнаружим, то язык L пуст.

Второй алгоритм использует диаграмму переходов автомата M . Вопрос о пустоте языка $L(M)$ эквивалентен вопросу о существовании пути из вершины

s_M в одну из вершин множества F_M . А этот вопрос алгоритмически разрешим. \square

Еще одной традиционно рассматриваемой алгоритмической проблемой для языков является проблема пустоты пересечения двух языков.

Проблема пустоты пересечения: по произвольным двум языкам L_1 и L_2 в алфавите Σ требуется определить, пусто ли их пересечение.

Теорема 5.3. Проблема пустоты пересечения для класса конечно-автоматных (регулярных) языков алгоритмически разрешима.

Доказательство. Если конечно-автоматные языки L_1 и L_2 распознаются конечными детерминированными автоматами M_1 и M_2 , то построим конечный автомат M , распознающий язык $L(M) = L_1 \cap L_2$ и решим для языка $L(M)$ проблему пустоты. \square

Проблема включения: по произвольным двум языкам L_1 и L_2 в алфавите Σ требуется определить, верно ли, что $L_1 \subseteq L_2$.

Теорема 5.4. Проблема включения для класса конечно-автоматных (регулярных) языков алгоритмически разрешима.

Доказательство. Разрешимость проблемы включения следует из следующей эквивалентности

$$L_1 \subseteq L_2 \iff L_1 \cap (\Sigma^* \setminus L_2) = \emptyset.$$

\square

Проблема совпадения языков: по произвольным двум языкам L_1 и L_2 в алфавите Σ требуется определить, совпадают ли они.

Теорема 5.5. Проблема совпадения для класса конечно-автоматных (регулярных) языков алгоритмически разрешима.

Доказательство. Разрешимость проблемы включения следует из следующей эквивалентности

$$L_1 = L_2 \iff (L_1 \subseteq L_2) \& (L_2 \subseteq L_1).$$

\square

Специфической алгоритмической проблемой для языков является проблема их бесконечности.

Проблема бесконечности языка: по произвольному языку L в алфавите Σ требуется определить, является ли язык L бесконечным.

Теорема 5.6. Проблема бесконечности для класса конечно-автоматных (регулярных) языков алгоритмически разрешима.

Доказательство. Используя теорему о накачке установим следующий критерий бесконечности языка L , распознаваемого конечным автоматом с n состояниями:

язык L бесконечен тогда и только тогда, когда он содержит такое слово w , что $n \leq |w| < 2n$.

Если язык L бесконечен, то он содержит слово u длины $|u| \geq 2n$. По теореме о накачке слово u можно представить в виде $u = xyz$, где $0 < |y| \leq n$ и $xz \in L$. Так как $|xz| \geq n$, то повторяя процесс, получим слово w такое, что $n \leq |w| < 2n$ и $w \in L$.

Обратно, если существует слово w такое, что $n \leq |w| < 2n$ и $w \in L$, то по теореме о накачке слово u можно представить в виде $u = xyz$, где $0 < |y|$ и при любом $k \geq 0$ $xy^kz \in L$, значит L – бесконечный язык.

Другой критерий бесконечности языка дается эквивалентностью

язык L является бесконечным тогда и только тогда, когда в его диаграмме переходов есть цикл. \square

§6. Минимизация автоматов

Конечные автоматы – весьма полезный аппарат для описания и изучения языков, компьютеров и алгоритмов. Так как один и тот же автоматный язык может быть задан различными конечными автоматами, то представляет интерес вопрос о *минимальном числе состояний* детерминированного конечного автомата, задающего этот же язык, т.е. эквивалентного данному конечному автомату.

Простейшая оптимизация состоит в удалении всех состояний, недостижимых из начального состояния, и всех ведущих к ним и из них переходов. Все достижимые состояния конечного автомата можно найти за полиномиальное время алгоритмом типа "поиск в ширину": *множество всех достижимых состояний конечного автомата M – это транзитивное замыкание одноэлементного множества $\{s_M\}$ относительно отношения*

$$\{ \langle p, \delta(p, a) \rangle \mid a \in \Sigma \}.$$

После удаления всех недостижимых состояний в полученном автомате все еще могут быть "лишние" состояния. Например, состояния, обладающие следующим свойством: исходя из любого из них при движении в заключительные состояния можно прочесть одни и те же слова. Такие состояния называются *эквивалентными*. Эквивалентные состояния можно "склеить", и полученный конечный автомат будет распознавать тот же самый язык, что и исходный автомат.

Определение 6.1. *Внутренние состояния q и r автомата M с входным алфавитом Σ_M называются эквивалентными, если для любого слова w в этом*

алфавите имеет место эквивалентность:

$$\delta_M^*(q, w) \in F_M \iff \delta_M^*(p, w) \in F_M.$$

Введенное отношение на множестве Q_M всех внутренних состояний автомата M является отношением эквивалентности, т.е. оно рефлексивно, симметрично и транзитивно. Будем его временно обозначать через \equiv . Обозначим через $[q]$ класс эквивалентности, определяемый внутренним состоянием q , а через $[Q_M]$ – множество всех классов эквивалентности, т.е. фактормножество множества Q_M по рассматриваемому отношению эквивалентности.

Построим новый автомат $[M]$ – факторавтомат автомата M по отношению эквивалентности \equiv , полагая

$$[M] = \langle \Sigma_{[M]}, Q_{[M]}, \delta_{[M]}, s_{[M]}, F_{[M]} \rangle,$$

где

$$\begin{aligned} \Sigma_{[M]} &= \Sigma_M, \quad Q_{[M]} = \{ [q] \mid q \in Q_M \}, \\ \delta_{[M]}([q], a) &= [\delta_M(q, a)], \quad s_{[M]} = [s_M], \quad \text{quad} F_{[M]} = \{ [q] \mid q \in F_M \}. \end{aligned}$$

Нетрудно проверить, что определения функции $\delta_{[M]}$ и множества $F_{[M]}$ корректны, т.е. не зависят от выбора представителей в классах эквивалентности, языки, принимаемые автоматами M и $[M]$, совпадают. Автоматы M и $[M]$ эквивалентны.

Для языка L в алфавите Σ , т.е. подмножества $L \subseteq \Sigma^*$, средствами самого этого языка определим на нем отношение эквивалентности \equiv_L , полагая для произвольных двух слов W и U из Σ^*

$$W \equiv_L U \iff \text{для любого слова } V \text{ из } \Sigma^* (WV \in L \iff UV \in L).$$

Отношение \equiv_L действительно является отношением эквивалентности, т.е. оно рефлексивно, симметрично и транзитивно. Отношение \equiv_L называется **эквивалентностью слов относительно языка L** .

Заметим, что если L – автоматный язык, принимаемый детерминированным автоматом M , то для произвольных двух слов W и U в его входном алфавите Σ справедлива эквивалентность:

W и U эквивалентны относительно языка L тогда и только тогда, когда состояния $\delta_M^(s_M, W)$ и $\delta_M^*(s_M, U)$ эквивалентны.*

Для языка L , принимаемого конечным детерминированным автоматом $M = \langle \Sigma_M, Q_M, \delta_M, s_M, F_M \rangle$, введем еще одно отношение эквивалентности \approx_M , полагая для произвольных слов W и U в алфавите Σ_M этого автомата:

$$W \approx_M U \iff \delta_M^*(s_M, W) = \delta_M^*(s_M, U).$$

Классу эквивалентности $[W]_{\approx_M}$ можно естественным образом сопоставить достижимое из s_M состояние $\delta_M^*(s_M, W)$, которое определено корректно, т.е. не зависит от выбора представителя в классе. Так как это соответствие является взаимнооднозначным, то число классов эквивалентности по отношению эквивалентности \approx_M равно числу достижимых из s_M состояний.

Следующая теорема устанавливает связь между двумя отношениями эквивалентности \approx_M и $\equiv_{L(M)}$.

Теорема 6.1. *Для произвольного детерминированного автомата M и любых слов W и U в его входном алфавите Σ_M из того, что $W \approx_M U$ следует, что $\equiv_{L(M)}$.*

Доказательство. Если $W \approx_M U$, то $\delta_M^*(s_M, W) = \delta_M^*(s_M, U)$. Поэтому для любого слова V во входном алфавите автомата M выполняются равенства

$$\delta_M^*(s_M, WV) = \delta_M^*(\delta_M^*(s_M, W), V) = \delta_M^*(\delta_M^*(s_M, U), V) = \delta_M^*(s_M, UV).$$

Значит, для любого слова V во входном алфавите автомата M справедлива эквивалентность:

$WV \in L(M)$ тогда и только тогда, когда $UV \in L(M)$.

Поэтому $W \equiv_{L(M)} U$. □

В силу доказанной теоремы получаем, что каждый класс эквивалентности относительно отношения \approx_M содержится в некотором классе эквивалентности относительно отношения $\equiv_{L(M)}$. Поэтому число классов эквивалентности относительно отношения $\equiv_{L(M)}$ не превосходит числа классов эквивалентности относительно отношения \approx_M .

Эта теорема дает важную информацию относительно любого автомата M , распознающего язык L :

число его внутренних состояний не меньше, чем число классов эквивалентности относительно \equiv_L .

Таким образом, число классов эквивалентности отношения \equiv_L служит естественной нижней границей для числа внутренних состояний любого автомата M , распознающего язык L .

Следующая теорема утверждает, что эта нижняя граница достижима.

Теорема 6.2. *Для любого регулярного языка L существует детерминированный конечный автомат с числом внутренних состояний, равным числу классов эквивалентности отношения \equiv_L , распознающий этот язык L .*

Доказательство. Обозначим через $[w]$ класс эквивалентности отношения \equiv_L с представителем w . По языку L в алфавите Σ построим детерминированный конечный автомат $M = \langle \Sigma, Q_M, \delta_M, s_M, F_M \rangle$, распознающий в точности язык L .

Полагаем:

$$\begin{aligned} Q_M &= \{ [w] \mid w \in \Sigma^* \} - \text{множество классов эквивалентности} \\ &\text{отношения } \equiv_L; \\ s_M &= [\varepsilon] - \text{класс эквивалентности, определяемый пустым словом;} \\ F_M &= \{ [u] \mid u \in L \}. \end{aligned}$$

Для произвольных $[w] \in Q_M$ и $a \in \Sigma$ полагаем $\delta_M([w], a) = [wa]$.

Так как язык L является регулярным, то он распознается некоторым детерминированным конечным автоматом M' . Число классов эквивалентности относительно отношения $\equiv_{L(M')}$ не превосходит числа классов эквивалентности относительно отношения $\approx_{M'}$. В свою очередь число классов эквивалентности по отношению эквивалентности $\approx_{M'}$, равное числу достижимых из $s_{M'}$ состояний, не превосходит число внутренних состояний конечного автомата M' .

Значит M – конечный автомат.

Для доказательства корректности определения функции δ_M равенством $\delta_M([w], a) = [wa]$ заметим, что если $w \equiv u$, то $wa \equiv ua$.

Покажем, что $L = L(M)$.

Индукцией по длине $|u|$ слова u покажем, что для любых слов w и u из Σ^* $[w]u \vdash_M^* [wu]\varepsilon$. Если u – пустое слово, то утверждение верно. Индуктивный переход происходит по схеме

$$[w]au \vdash_M [wa]u \vdash_M^* [(wa)u]\varepsilon = [w(au)]\varepsilon.$$

Таким образом $\delta_M^*([w], u) = [wu]$.

Для завершения доказательства воспользуемся равенством:

$\delta_M^*([\varepsilon], u) = [\varepsilon u] = [u]$ и эквивалентностями

$$u \in L(M) \iff \delta_M^*([\varepsilon], u) \in F_M \iff [u] \in F_M \iff u \in L.$$

Поэтому $L(M) = L$. □

Замечание. Построенный в ходе доказательства теоремы по языку L автомат M носит название *стандартный автомат для L* и обозначается через M_L .

Теорема 6.3 (Майхилл – Нероуд). *Язык L является регулярным тогда и только тогда, когда конечно число классов эквивалентности отношения \equiv_L .*

До к а з а т е л ь с т в о. Если язык L является регулярным, то он распознается некоторым детерминированным конечным автоматом M с числом состояний не меньшим, чем число классов эквивалентности отношения \equiv_L . Поэтому у отношения \equiv_L конечное число классов эквивалентности.

Обратно, если у отношения \equiv_L конечное число классов эквивалентности, то стандартный детерминированный конечный автомат M_L распознает язык L . □

В качестве примера применения теоремы Майхилла – Нероуда рассмотрим еще одно доказательство нерегулярности языка

$$L_1 = \{a^n b^n | n \geq 1\}.$$

При $i \neq j$ слова a^i и a^j не являются эквивалентными относительно \equiv_L , так как слово b^i , будучи записанным после a^i , дает слово из L_1 , а будучи записанным после a^j дает слово, не входящее в L_1 . Значит существует бесконечно много классов эквивалентности $[a]$, $[a^2]$, $[a^3]$, ... Поэтому язык L_1 не является регулярным.

Вернемся еще раз к вопросу об эквивалентности состояний конечного автомата $M = \langle \Sigma, Q_M, \delta_M, s_M, F_M \rangle$.

Для произвольного его внутреннего состояния q рассмотрим следующий язык

$$L_q = \{w | \delta_M^*(q, w) \in F_M\}.$$

Тогда вопрос об эквивалентности состояний q и p равносильен вопросу о совпадении языков L_q и L_p . Как уже отмечалось выше, этот вопрос алгоритмически разрешим.

Рассмотрим некоторое достаточно естественное обобщение этой ситуации. Пусть L – некоторый язык в алфавите Σ .

Определение 6.2. Внутренние состояния p и q конечного автомата $M = \langle \Sigma, Q_M, \delta_M, s_M, F_M \rangle$ называются **эквивалентными относительно языка L** в алфавите Σ , если для любого слова w языка L имеет место эквивалентность:

$$\delta_M^*(p, w) \in F_M \text{ тогда и только тогда, когда } \delta_M^*(q, w) \in F_M.$$

Нетрудно понять, что внутренние состояния p и q конечного автомата $M = \langle \Sigma, Q_M, \delta_M, s_M, F_M \rangle$ являются **эквивалентными относительно языка L** в алфавите Σ тогда и только тогда, когда выполняется равенство $L_p \cap L = L_q \cap L$.

Теорема 6.4. Существует алгоритм, позволяющий для любых двух внутренних состояний p и q произвольного конечного автомата

$$M = \langle \Sigma, Q_M, \delta_M, s_M, F_M \rangle$$

и любого конечно-автоматного языка L ответить на вопрос, эквивалентны ли состояния p и q относительно языка L .

Д о к а з а т е л ь с т в о. Так как при сделанных в условии теоремы предположениях языки L_p , L_q и L являются конечно-автоматными, то таковы же и языки $L_p \cap L$ и $L_q \cap L$. Поэтому вопрос об их совпадении алгоритмически разрешим. \square

Если несколько обобщить ситуацию и выйти из класса конечно-автоматных языков L в несколько более широкий, но не слишком, класс языков с конечными заданиями, то проблема эквивалентности внутренних состояний автоматов относительно языков L этого класса оказывается алгоритмически неразрешимой.

Не претендуя на полноту изложения, наметим основные этапы доказательства этого утверждения.

Обозначим через N_0 множество неотрицательных целых чисел, т.е. натуральных чисел вместе с нулем.

Можно доказать (это будет сделано позже), что для натурального числа n существует вычислимая функция $c(x_1, \dots, x_n)$ от n переменных, принимающих значения из N_0 , и n вычислимых функций от одной переменной $c_{1,n}(t), \dots, c_{n,n}(t)$, причем все эти функции "имеют конечные задания" и выполняются равенства

$$c(c_{1,n}(t), \dots, c_{n,n}(t)) = t, \\ c_{1,n}(c(x_1, \dots, x_n)) = x_1, \dots, c_{n,n}(c(x_1, \dots, x_n)) = x_n.$$

Функция $c(x_1, \dots, x_n)$ осуществляет нумерацию n -ок чисел из N_0 .

Следующая фундаментальная теорема, доказанная российским математиком Ю.В. Матиясевичем [48] в конце 60-х годов XX века, завершила исследования по 10-ой проблеме Д. Гильберта, начатые в 50-е годы XX века зарубежными математиками М. Дэвисом, Х. Путнамом и Дж. Робинсоном, и является одним из самых выдающихся математических результатов третьей четверти XX века.

Теорема 6.5 (Ю.В. Матиясевич). *Невозможно создать алгоритм, позволяющий по произвольному полиному $F(x_1, \dots, x_n)$ с целыми коэффициентами ответить на вопрос, имеет ли уравнение*

$$F(x_1, \dots, x_n) = 0$$

решение в целых неотрицательных числах.

Замечание. Речь идет о решении уравнения в целых неотрицательных числах по чисто техническим причинам. Аналогичный результат справедлив и для решений в целых или в натуральных числах.

Рассмотрим функцию $sg(x)$, равную 0 при $x = 0$ и 1 при $x > 0$. Полагаем $\varphi(t) = sg(|F(c_{1,n}(t), \dots, c_{n,n}(t))|)$.

Определим язык

$$L_\varphi = \{ \varphi(0)\varphi(1) \dots \varphi(n) \mid n \geq 0 \}$$

в алфавите $\{0, 1\}$.

Введем в рассмотрение конечный автомат

$$M = \langle \{0, 1\}, \{q_1, q_2, q_3\}, q_1, \delta, \{q_3\} \rangle,$$

функция переходов δ которого задана правилами

$$q_1 1 \rightarrow q_1, q_1 0 \rightarrow q_3, q_2 1 \rightarrow q_2, q_2 0 \rightarrow q_2, q_3 1 \rightarrow q_3, q_3 0 \rightarrow q_3.$$

Нетрудно понять, что для произвольного языка L в алфавите $\{0, 1\}$ имеет место эквивалентность:

состояния q_1 и q_2 автомата M эквивалентны относительно языка L тогда и только тогда, когда $L \subseteq L(1^)$.*

Поэтому

уравнение $F(x_1, \dots, x_n) = 0$ не имеет решений в целых неотрицательных числах тогда и только тогда, когда состояния q_1 и q_2 автомата M эквивалентны относительно языка L_φ .

Так как первый вопрос алгоритмически неразрешим, то тем самым установлена справедливость следующей теоремы.

Теорема 6.6. *Невозможно создать алгоритм, позволяющий для внутренних состояний q_1 и q_2 конечного автомата*

$$M = \langle \{0, 1\}, \{q_1, q_2, q_3\}, q_1, \delta, \{q_3\} \rangle$$

с функцией переходов δ , заданой правилами

$$q_1 1 \rightarrow q_1, q_1 0 \rightarrow q_3, q_2 1 \rightarrow q_2, q_2 0 \rightarrow q_2, q_3 1 \rightarrow q_3, q_3 0 \rightarrow q_3,$$

и произвольного языка L_φ ответить на вопрос, эквивалентны ли состояния q_1 и q_2 автомата M относительно языка L_φ .

Для произвольного языка L в алфавите Σ , т.е. подмножества $L \subseteq \Sigma^*$, средствами самого этого языка определим на нем еще одно отношение эквивалентности \sim_L , полагая для произвольных двух слов W и U из Σ^*

$$W \sim_L U \iff$$

$$\text{для любых двух слов } X \text{ и } Y \text{ из } \Sigma^* (XWY \in L \iff XUY \in L).$$

Отношение \sim_L действительно является отношением эквивалентности, т.е. оно рефлексивно, симметрично и транзитивно. Более того, отношение \sim_L является конгруэнтностью на свободном моноиде Σ^* , т.е. если $W_1 \sim_L W_2$ и $U_1 \sim_L U_2$, то $W_1 U_1 \sim_L W_2 U_2$.

Это позволяет на множестве классов эквивалентности $[W]_{\sim_L}$ естественным образом ввести операцию умножения $*$, полагая

$$[W]_{\sim_L} * [U]_{\sim_L} = [WU]_{\sim_L}.$$

Легко проверяется корректность определения, т.е. его независимость от выбора представителей в классах эквивалентности. Значит, множество классов эквивалентности $[W]_{\sim_L}$ относительно операции умножения $*$ является моноидом.

Заметим, что если L – автоматный язык, принимаемый детерминированным автоматом M , то для произвольных двух слов W и U в его входном алфавите Σ справедлива эквивалентность:

$W \sim_L U$ тогда и только тогда, когда для любого достижимого из s_M состояния q эквивалентны состояния $\delta_M^*(q, W)$ и $\delta_M^*(q, U)$.

Можно было бы пойти несколько иным путем. Каждому слову w в алфавите Σ сопоставим множество $C_L(w)$ всех его контекстов относительно языка L , полагая

$$C_L(w) = \{ \langle x, y \rangle \mid xwy \in L \}.$$

Множество $\{ C_L(w) \mid w \in \Sigma^* \}$ можно наделить структурой моноида, определив на нем операцию \star равенством

$$C_L(w) \star C_L(u) = C_L(wu).$$

Нетрудно проверить, что отображение $[W]_{\sim_L} \mapsto C_L(W)$ задает изоморфизм этих двух моноидов, называемых синтаксическими моноидами языка L .

§7. Конечные автоматы с выходом

Мы рассматривали автоматы в качестве распознавателей языков, однако они могут выполнять и другие функции, например, выступать в качестве математических моделей дискретных преобразователей.

В этом параграфе речь пойдет об автоматах с выходом, называемых автоматами Мили.

У автомата Мили, который мы будем называть просто автоматом, имеются *входная* и *выходная* ленты, разбитые на ячейки, в каждой из которых можно записывать по одному символу из входного и соответственно выходного алфавита. Слова во входном алфавите автомата записываются побуквенно в ячейки входной ленты. Основная часть конечного автомата – *считывающая головка*. Она представляет собой нечто вроде черного ящика и в каждый конкретный момент времени может находиться в одном из конечного числа различных *внутренних состояний* автомата. Считывающая головка может распознать, какой символ записан в обозреваемой ячейке входной ленты. Первоначально считывающая головка размещается на крайнем левом краю ленты и приводится в некоторое внутренне состояние. Работа автомата осуществляется по тактам. На очередном такте работы автомата считывающая головка читает символ из обозреваемой ячейки входной ленты, переходит в новое состояние, зависящее лишь от текущего состояния и только что прочитанного символа, печатает на выходной ленте символ и сдвигается в соседнюю справа ячейку. Так как новое состояние автомата и помещаемый на выходной ленте символ однозначно определены, то он называется *детерминированным* автоматом. Процесс повторяется, пока не будет прочитано все входное слово.

Переходим к формальному определению детерминированного автомата с выходом.

Определение 7.1. *Детерминированный автомат с выходом* – это пятерка

$$M = \langle \Sigma_M, Q_M, \Omega_M, \delta_M, \rho_M \rangle,$$

где

Σ_M – *входной алфавит* автомата M ,

Q_M – *внутренний алфавит* автомата M , его элементы называются *внутренними состояниями*,

Ω_M – *выходной алфавит* автомата M ,

δ_M – *функция переходов* автомата M , это функция из $Q_M \times \Sigma_M$ в Q_M ,

ρ_M – *функция выходов* автомата M , это функция из $Q_M \times \Sigma_M$ в Ω_M .

Автомат M называется конечным, если конечны его алфавиты Σ_M , Q_M и Ω_M .

В ячейки входной ленты автомата M записываются буквы входного алфавита Σ_M , таким образом на ленте оказывается записанное слово. Автомат функционирует в дискретные моменты времени (такты). Если в t -м такте считывающая головка находится в некотором внутреннем состоянии $q_t \in Q_M$, а в обозреваемой ею ячейке находится (прочитывается головкой) символ $a_t \in \Sigma_M$, то автомат печатает на выходной ленте выходной символ $b_t = \rho_M(q_t, a_t) \in \Omega_M$, головка переходит в состояние $q_{t+1} = \delta_M(q_t, a_t) \in Q_M$ и сдвигается вправо. Следующие равенства задают закон функционирования автомата во времени и называются каноническими уравнениями автомата

$$b_t = \rho_M(q_t, a_t), \quad q_{t+1} = \delta_M(q_t, a_t).$$

Если в начальный момент на входной ленте автомата M записано слово $w = w_1 w_2 \dots w_n$, считывающая головка обозревает самую левую ячейку ленты (в которой записана первая буква слова w) и приведена в некоторое начальное состояние q_1 , то, производя вычисления при данном входе, автомат напечатает на выходной ленте слово $u = u_1 u_2 \dots u_n$ той же длины n , что и входное слово, пройдя последовательность внутренних состояний $q_1 q_2 q_3 \dots q_{n+1}$.

При этом говорят, что автомат M **преобразовал** входное слово (входную последовательность) $w = w_1 w_2 \dots w_n$ в выходное слово (выходную последовательность) $u = u_1 u_2 \dots u_n$. В некоторых случаях мы будем говорить, что входное слово (входная последовательность) $w = w_1 w_2 \dots w_n$ переводит состояние q_1 в состояние q_{t+1} .

Естественным образом продолжим функции ρ_M и δ_M до функций

$$\rho_M^* : Q_M \times \Sigma_M^* \rightarrow \Omega_M^*, \quad \delta_M^* : Q_M \times \Sigma_M^* \rightarrow Q_M^*,$$

полагая для любого внутреннего состояния $q \in Q_M$

$$\rho_M^*(q, \varepsilon) = \varepsilon, \quad \delta_M^*(q, \varepsilon) = q,$$

и для любого символа a входного алфавита Σ_M и любого слова u в этом алфавите

$$\delta_M^*(q, ua) = \delta_M(\delta_M^*(q, u), a), \quad \rho_M^*(q, ua) = \rho_M^*(q, u), a) \rho_M(\delta_M^*(q, u), a).$$

Индукцией по длине слов нетрудно доказать, что для любых двух слов w и u во входном алфавите Σ_M и любого внутреннего состояния $q \in Q_M$ выполняются равенства

$$\delta_M^*(q, wi) = \delta_M^*(\delta_M^*(q, w), u), \quad \rho_M^*(q, wi) = \rho_M^*(q, w), a) \rho_M^*(\delta_M^*(q, w), u).$$

При каждом фиксированном внутреннем состоянии q получаем отображение

$$M_q : \Sigma_M^* \rightarrow \Omega_M^*,$$

задаваемое равенством $M_q(w) = \rho_M^*(q, w)$, которое называется **автоматным отображением, соответствующим внутреннему состоянию q** .

Для задания детерминированного конечного автомата с выходом

$$M = \langle \Sigma_M, Q_M, \Omega_M, \delta_M, \rho_M \rangle,$$

необходимо задать

входной алфавит Σ_M ,

внутренний алфавит Q_M ,

выходной алфавит Ω_M ,

функцию переходов δ_M и

функцию выходов ρ_M .

Это можно сделать, например, с помощью двух таблиц, строки которых помечены элементами внутреннего алфавита Q_M , а столбцы – элементами входного алфавита Σ_M .

На пересечении строки, помеченной состоянием q , и столбца, помеченного символом a , в первой таблице стоит $\delta_M(q, a)$, а во второй – $\rho_M(q, a)$. Нетрудно оценить число таких пар таблиц, а значит, и число различных автоматов Мили, оно равно $(mn)^{rn}$, где $r = |\Sigma_M|$, $n = |Q_M|$ и $m = |\Omega_M|$.

Однако для задания автомата с выходом удобнее использовать более наглядное графическое представление, аналогичное диаграмме состояний или диаграмме переходов, уже использовавшейся нами для задания автоматов без выхода. Этой цели будет служить размеченный ориентированный граф, вершины которого соответствуют состояниям автомата и ими помечены, а дуги графа соответствуют переходам между состояниями и несут информацию о выходных символах. Более точно, *из вершины, помеченной состоянием q , берет начало дуга, заканчивающаяся в вершине, помеченной состоянием p , и сама помеченная словом a/b при условии, что $\delta_M(q, a) = p$ и $\rho_M(q, a) = b$* .

Одним из важнейших понятий теории автоматов является понятие **эквивалентности состояний** автомата с выходом.

Определение 7.2. Состояния p и q автомата M называются **эквивалентными**, если соответствующие им автоматные отображения M_p и M_q совпадают, т.е. если для любого слова w во входном алфавите Σ_M автомата M выполняется равенство $\rho_M^*(p, w) = \rho_M^*(q, w)$.

Запись $p \sim q$ будет выражать эквивалентность состояний p и q .

Через Q_M / \sim обозначим фактормножество множества Q_M внутренних состояний автомата M по отношению эквивалентности \sim , т.е. множество соответствующих классов эквивалентности $[p]_{\sim}$.

Рассмотрим новый автомат

$$M / \sim = \langle \Sigma_M, Q_M / \sim, \Omega_M, \delta_M / \sim, \rho_M / \sim \rangle,$$

где

Σ_M – **входной алфавит** автомата M , являющийся входным алфавитом и автомата M / \sim ,

Q_M / \sim – **внутренний алфавит** автомата M / \sim , являющийся фактормножеством множества Q_M внутренних состояний автомата M по отношению эквивалентности \sim ,

Ω_M – **выходной алфавит** автомата M , являющийся выходным алфавитом и автомата M / \sim ,

δ_M / \sim – **функция переходов** автомата M / \sim и

ρ_M / \sim – **функция выходов** автомата M / \sim , заданные равенствами

$$\rho_M / \sim ([q], a) = \rho_M(q, a), \quad \delta_M / \sim ([q], a) = [\delta_M(q, a)].$$

Корректность определений, т.е. независимость от выбора представителей, легко проверяется.

Рассмотрим вопрос об алгоритмической распознаваемости эквивалентности состояний конечного автомата с выходом.

Теорема 7.1 (Хаффман – Мили). *Если f и g – внутренние состояния автомата Мили M с n внутренними состояниями и для любого слова w длины $|w| = n - 1$ выполняется равенство $\rho_M^*(f, w) = \rho_M^*(g, w)$, то состояния f и g эквивалентны.*

До к а з а т е л ь с т в о. Для произвольного целого неотрицательного числа k введем понятие k -эквивалентности внутренних состояний автомата: внутренние состояния p и q автомата Мили M называются k -эквивалентными, если для любого слова w длины $|w| = k$ выполняется равенство $\rho_M^*(p, w) = \rho_M^*(q, w)$. k -эквивалентность внутренних состояний p и q будем обозначать через $p \sim_k q$.

Заметим, что любые два внутренних состояния p и q 0-эквивалентны.

Отношение k -эквивалентности является отношением эквивалентности, т.е. оно рефлексивно, симметрично и транзитивно.

Кроме того, если $p \sim_{k+1} q$, то $p \sim_k q$. Поэтому $[p]_{\sim_{k+1}} \subseteq [p]_{\sim_k}$, значит, $|Q_M / \sim_{k+1}| \geq |Q_M / \sim_k|$.

Если $|Q_M / \sim_{k+1}| = |Q_M / \sim_k|$, то каждое включение является на самом деле равенством $[p]_{\sim_{k+1}} = [p]_{\sim_k}$.

Покажем, что если для любого состояния s выполняется равенство $[s]_{\sim_{k+1}} = [s]_{\sim_k}$, то для любого состояния p выполняется равенство $[p]_{\sim_{k+2}} = [p]_{\sim_{k+1}}$, а значит и равенство $[p]_{\sim_{k+2}} = [p]_{\sim_k}$.

Заметим, что $[p]_{\sim_{k+2}} \subseteq [p]_{\sim_{k+1}}$.

Для доказательства обратного включения, предположим, что $p \sim_{k+1} q$. Тогда для любого слова w длины $|w| = k$ и любого символа a входного алфавита выполняется равенство $\rho_M^*(p, aw) = \rho_M^*(q, aw)$, т.е. равенство

$$\rho_M(p, a)\rho_M^*(\delta_M(p, a), w) = \rho_M(q, a)\rho_M^*(\delta_M(q, a), w).$$

Значит

$$\rho_M(p, a) = \rho_M(q, a), \quad \text{и} \quad \rho_M^*(\delta_M(p, a), w) = \rho_M^*(\delta_M(q, a), w).$$

Поэтому состояния $\delta_M(p, a)$ и $\delta_M(q, a)$ k -эквивалентны, значит они и $(k+1)$ -эквивалентны. Следовательно, для любого символа b входного алфавита выполняется равенство

$$\rho_M^*(\delta_M(p, a), wb) = \rho_M^*(\delta_M(q, a), wb).$$

Значит, верно равенство

$$\rho_M(p, a)\rho_M^*(\delta_M(p, a), wb) = \rho_M(q, a)\rho_M^*(\delta_M(q, a), wb),$$

т.е. равенство

$$\rho_M^*(p, awb) = \rho_M^*(q, awb).$$

Так как при сделанных предположениях awb – произвольное слово длины $k+2$, то $p \sim_{k+2} q$. Поэтому $[p]_{\sim_{k+1}} \subseteq [p]_{\sim_{k+2}}$. А значит $[p]_{\sim_{k+1}} = [p]_{\sim_{k+2}}$.

Отсюда сразу следует, что если для всех состояний p выполнено равенство $[p]_{\sim_{k+1}} = [p]_{\sim_k}$, то при любом $t > k$ для всех состояний p выполняется равенство $[p]_{\sim_t} = [p]_{\sim_k}$.

Рассмотрим такое k , что

$$1 = |Q_M/\sim_0| < |Q_M/\sim_1| < \dots < |Q_M/\sim_{k-1}| < |Q_M/\sim_k| = |Q_M/\sim_{k+1}|.$$

Тогда $k+1 \leq |Q_M/\sim_k| \leq n$, поэтому $k \leq n-1$. Значит $(n-1)$ -эквивалентность внутренних состояний p и q влечет их k -эквивалентность, а значит, и просто эквивалентность.

Так как по условию теоремы состояния f и g $(n-1)$ -эквивалентны, то они k -эквивалентны, а значит, и просто эквивалентны. \square

Из доказанной теоремы получаем *алгоритм, позволяющий по любым двум внутренним состояниям q и p автомата Мили M с n внутренними состояниями ответить на вопрос, эквивалентны ли эти состояния*. Алгоритм заключается в проверке для всех слов w длины $|w| = n-1$ выполнимости равенства

$$\rho_M^*(p, w) = \rho_M^*(q, w).$$

Для автоматов с выходом представляет интерес *вопрос о различимости входных слов*.

Определение 7.3. Слова u и v во входном алфавите Σ автомата M называются неразличимыми автоматом M в начальном внутреннем состоянии q , если для любого слова w во входном алфавите автомата M выполняется равенство

$$\rho_M^*(q, u \cdot w) = \rho_M^*(q, v \cdot w).$$

Если слова u и v во входном алфавите Σ автомата M неразличимы автоматом M в любом начальном внутреннем состоянии, то они называются неразличимыми автоматом M .

Теорема 7.2. Слова u и v во входном алфавите Σ автомата M неразличимы автоматом M в начальном внутреннем состоянии q тогда и только тогда, когда выполняется равенство

$$\rho_M^*(q, u) = \rho_M^*(q, v)$$

и эквивалентны состояния $\delta_M^*(q, u)$ и $\delta_M^*(q, v)$ автомата M .

Доказательство легко следует из равенств

$$\begin{aligned} \rho_M^*(q, uw) &= \rho_M^*(q, u), a) \rho_M^*(\delta_M^*(q, u), w), \\ \rho_M^*(q, vw) &= \rho_M^*(q, v), a) \rho_M^*(\delta_M^*(q, v), w). \end{aligned}$$

□

Следствие. Существует алгоритм, позволяющий по произвольному конечному автомату M и по любым двум словам в его входном алфавите определить, различимы ли они автоматом M .

Не вдаваясь в подробности, рассмотрим вопрос, связанный с периодичностью в конечных автоматах.

Если на вход конечного автомата с выходом $M = \langle \Sigma_M, Q_M, \Omega_M, \delta_M, \rho_M \rangle$ при начальном состоянии q подавать посимвольно бесконечную последовательность

$$\bar{w} = \{w_1, w_2, \dots, w_n, \dots\}$$

букв его входного алфавита Σ_M (такие последовательности называются *супер-словами* в алфавите Σ_M), то на выходе будет вырабатываться посимвольно бесконечная последовательность

$$\bar{u} = \{u_1, u_2, \dots, u_n, \dots\}$$

букв его выходного алфавита Ω_M (суперслово в выходном алфавите). Эту последовательность мы будем обозначать через $M_q(\bar{w})$. Закон ее образования задается равенствами

$$u_1 = \rho_M(q, w_1), \quad u_{n+1} = \rho_M(\delta_M^*(q, w_1 \dots w_n), w_{n+1}).$$

Таким образом, автомат M суперслова во входном алфавите Σ_M перерабатывает в суперслова в выходном алфавите Ω_M .

В этом случае будем говорить, что конечный автомат M под воздействием входной последовательности \bar{w} из начального состояния q проходит (вырабатывает) выходную последовательность $M_q(\bar{w})$.

Теорема 7.3. *Конечный автомат M под воздействием периодической входной последовательности \bar{w} , состоящей из букв его входного алфавита, с предпериодом N и периодом T из начального состояния q проходит периодическую последовательность состояний \bar{q} с предпериодом $N'(q)$ и периодом $T'(q)$ и вырабатывает периодическую выходную последовательность \bar{u} с предпериодом $N''(q)$ и периодом $T''(q)$, для которых выполняются неравенства*

$$N'(q), N''(q) \leq N + (d_q - 1)T,$$

$$T'(q) = k't', T''(q) = k''t'',$$

где d_q – число состояний автомата M , достижимых из состояния q , $t', t'' \leq d_q$, k' и k'' – делители T .

Доказательство. Полагаем

$$\bar{w} = \{w_1, w_2, \dots, w_n, \dots\},$$

$$\bar{u} = \{u_1, u_2, \dots, u_n, \dots\},$$

$$\bar{q} = \{q_1, q_2, \dots, q_n, \dots\},$$

где $q_1 q$. Мы ограничимся доказательством лишь периодичности последовательностей \bar{u} и \bar{q} .

Рассмотрим следующую подпоследовательность \bar{q}' последовательности \bar{q}

$$\bar{q}' = \{q_{N+1}, q_{N+1+T}, q_{N+1+2T}, q_{N+1+3T} \dots, q_{N+1+nT}, \dots\}.$$

Так как в последовательности \bar{q} не более чем d_q различных членов, то в подпоследовательности \bar{q}' имеются повторения. Пусть l – наименьшее натуральное число, для которого существует такое неотрицательное целое m , что $0 \leq m < l$ и $q_{N+1+mT} = q_{N+1+lT}$. Так как по условию входная последовательность \bar{w} является периодической с предпериодом N и периодом T , то, в частности, для любого j выполняется равенство $w_{N+1+mT+j} = w_{N+1+lT+j}$, а значит, и равенства $q_{N+1+mT+j} = q_{N+1+lT+j}$ и $u_{N+1+mT+j} = u_{N+1+lT+j}$. Поэтому \bar{q} и \bar{u} – периодические последовательности с периодом $(l - m)T$.

При этом $(l - m) \leq d_q$. Значит $(l - m)$ не превосходит числа внутренних состояний автомата M . \square

Для произвольного алфавита Δ через Δ^{**} будем обозначать множество всех суперслов в алфавите Δ , т.е. множество всех бесконечных последовательностей символов алфавита Δ .

Если зафиксировать внутреннее состояние $q \in Q_M$ конечного автомата

$$M = \langle \Sigma_M, Q_M, \Omega_M, \delta_M, \rho_M \rangle,$$

то мы получим инициальный конечный автомат

$$M_q = \langle \Sigma_M, Q_M, \Omega_M, q, \delta_M, \rho_M \rangle,$$

который способен вычислять отображения из Σ_M^* в Ω_M^* и из Σ_M^{**} в Ω_M^{**} .

Чтобы иметь возможность говорить о вычислимости n -местных отображений из Σ_M^{**} в Ω_M^{**} , достаточно ввести в рассмотрение автоматы с входным алфавитом вида Σ_M^n . При этом мы отождествляем последовательность

$$(\{w_{1,1}, w_{1,2}, \dots, w_{1,m}, \dots\}, \{w_{2,1}, w_{2,2}, \dots, w_{2,m}, \dots\}, \dots, \{w_{n,1}, w_{n,2}, \dots, w_{n,m}, \dots\})$$

с последовательностью

$$\{(w_{1,1}, w_{2,1}, \dots, w_{n,1}), (w_{1,2}, w_{2,2}, \dots, w_{n,2}), \dots, (w_{1,m}, w_{2,m}, \dots, w_{n,m}), \dots\}.$$

Тогда автомат, вычисляющий n -местное отображение из Σ_M^{**} в Ω_M^{**} , – это автомат вида

$$M = \langle (\Sigma_M)^n, Q_M, \Omega_M, \delta_M, \rho_M \rangle.$$

Для реализации вычисления суперпозиции автоматных функций нам потребуется по двум автоматам построить их суперпозицию.

Определение 7.4. *Суперпозицией (последовательным соединением) автоматов*

$$M_1 = \langle \Sigma, Q_1, \Omega, \delta_1, \rho_1 \rangle$$

и

$$M_2 = \langle \Omega, Q_2, \Delta, \delta_2, \rho_2 \rangle$$

называется автомат

$$M = \langle \Sigma, Q, \Delta, \delta, \rho \rangle,$$

для которого $Q = Q_1 \times Q_2$ и

$$\begin{aligned} \delta(q, a) &= \delta((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, \rho_1(q_1, a))), \\ \rho(q, a) &= \rho((q_1, q_2), a) = \rho_2(q_2, \rho_1(q_1, a)). \end{aligned}$$

Суперпозицию автоматов M_1 и M_2 будем обозначать через $M_1 \circ M_2$. При этом будем считать, что если M_1 и M_2 – инициальные автоматы с начальными состояниями s_1 и s_2 соответственно, то $M_1 \circ M_2$ – инициальный автомат с начальным состоянием (s_1, s_2) . Нетрудно понять, что если инициальный автомат

$$M_1 = \langle \Sigma, Q_1, \Omega, s_1, \delta_1, \rho_1 \rangle$$

вычисляет функцию f из Σ^* в Ω^* , а инициальный автомат

$$M_2 = \langle \Omega, Q_2, \Delta, s_2, \delta_2, \rho_2 \rangle$$

вычисляет функцию g из $\Omega^* \Delta^*$, то инициальный автомат

$$M_1 \circ M_2 = \langle \Sigma, Q, \Delta, (s_1, s_2), \delta, \rho \rangle,$$

вычисляет функцию $g \circ f$ из Σ^* в Δ^* , являющуюся суперпозицией функций f и g .

Аналогично, если инициальный автомат

$$M_1 = \langle \Sigma, Q_1, \Omega, s_1, \delta_1, \rho_1 \rangle$$

вычисляет функцию f из Σ^{**} в Ω^{**} , а инициальный автомат

$$M_2 = \langle \Omega, Q_2, \Delta, s_2, \delta_2, \rho_2 \rangle$$

вычисляет функцию g из $\Omega^{**} \Delta^{**}$, то инициальный автомат

$$M_1 \circ M_2 = \langle \Sigma, Q, \Delta, (s_1, s_2), \delta, \rho \rangle,$$

вычисляет функцию $g \circ f$ из Σ^{**} в Δ^{**} , являющуюся суперпозицией функций f и g .

Обобщим ситуацию. Пусть при любом i ($1 \leq i \leq m$) инициальный автомат

$$M_i = \langle \Sigma^n, Q_i, \Omega, s_i, \delta_i, \rho_i \rangle$$

вычисляет n -местную функцию f_i из $(\Sigma^n)^{**}$ в Ω^{**} , а инициальный автомат

$$M_0 = \langle \Omega^m, Q_0, \Delta, s_0, \delta_0, \rho_0 \rangle$$

вычисляет m -местную функцию f_0 из $(\Omega^m)^{**} \Delta^{**}$. Построим инициальный автомат

$$M = \langle \Sigma^n, Q, \Delta, ((s_1, s_2, \dots, s_m), s_0), \delta, \rho \rangle,$$

вычисляющий n -местную функцию

$$f_0(f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n))$$

из $(\Sigma^n)^{**}$ в Δ^{**} , являющуюся суперпозицией функций f_0 и f_1, \dots, f_m .

Для этого, прежде всего, построим прямое произведение $M_1 \times \dots \times M_m$ автоматов M_1, \dots, M_m , полагая

$$M_1 \times \dots \times M_m = \langle \Sigma^n, Q_1 \times \dots \times Q_m, \Omega^m, (s_1, s_2, \dots, s_m), \delta, \rho \rangle,$$

где

$$\begin{aligned} \delta((q_1, \dots, q_m), (a_1, \dots, a_n)) &= (\delta_1(q_1, (a_1, \dots, a_n)), \dots, \delta_m(q_m, (a_1, \dots, a_n))), \\ \rho((q_1, \dots, q_m), (a_1, \dots, a_n)) &= (\rho_1(q_1, (a_1, \dots, a_n)), \dots, \rho_m(q_m, (a_1, \dots, a_n))). \end{aligned}$$

А затем возьмем суперпозицию $(M_1 \times \dots \times M_m) \circ M_0$.

Если зафиксировать входной алфавит Σ и рассматривать лишь автоматные функции (от произвольного числа аргументов) из Σ^{**} в Σ^{**} , то из предыдущих рассуждений следует, что суперпозиция автоматных функций есть функция автоматная. Поэтому, как и для булевых функций, можно поставить вопрос о существовании конечных полных систем автоматных функций. Однако ситуация, в отличие от случая булевых функций, оказывается более сложной.

Теорема 7.4. *Если алфавит Σ содержит не менее двух букв, то для множества всех автоматных функций (от произвольного числа аргументов) из Σ^{**} в Σ^{**} не существует конечной системы автоматных функций, из которых с помощью операции суперпозиции можно было бы получить любую автоматную функцию.*

Предварительно докажем вспомогательную лемму.

Лемма 1. *Пусть автоматные функции $f_0(x_1, \dots, x_m)$, $f_1(x_1, \dots, x_n)$, \dots , $f_m(x_1, \dots, x_n)$ из Σ^{**} в Σ^{**} вычислимы конечными автоматами с n_0 , n_1 , \dots , n_m состояниями, $\max(n_0, n_1, \dots, n_m) \leq N$ и*

$$f(x_1, \dots, x_n) = f_0(f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)).$$

Если \bar{w} – периодическая последовательность с периодом T , все простые делители которого не превосходят N , то $f(\bar{w})$ – периодическая последовательность с периодом, все простые делители которого также не превосходят N .

Д о к а з а т е л ь с т в о. Пусть \bar{w} – периодическая последовательность с периодом T , все простые делители которого не превосходят N .

Тогда последовательности

$$u_1 = f_1(\bar{w}), \dots, u_m = f_m(\bar{w})$$

являются периодическими с периодами

$$T_1 = k_1 T, \dots, T_m = k_m T,$$

причем $k_1, \dots, k_m \leq N$. Поэтому каждый простой делитель любого из чисел T_1, \dots, T_m также не превосходит N .

Последовательность (u_1, \dots, u_m) является периодической с периодом T_0 , равным наименьшему общему кратному чисел T_1, \dots, T_m . Поэтому любой простой делитель числа T_0 также не превосходит N .

Последовательность $u_0 = f_0(u_1, \dots, u_m)$ является периодической с периодом $T^* = k_0 T_0$, где $k_0 \leq N$.

Поэтому любой простой делитель числа T^* также не превосходит N . \square

Д о к а з а т е л ь с т в о теоремы. Предположим противное. Пусть f_1, \dots, f_m – конечная система автоматных функций из Σ^{**} в Σ^{**} , из которой с помощью операции суперпозиции можно получить любую автоматную функцию из

Σ^{**} в Σ^{**} . Если M_1, \dots, M_m – конечные автоматы с n_1, \dots, n_m внутренними состояниями, вычисляющие функции f_1, \dots, f_m , то выберем *простое число* $p > N = \max(n_1, \dots, n_m)$.

Пусть a и b – два различных элемента алфавита Σ . Обозначим через

$$M^{(p)} = \langle \Sigma, Q, \Sigma, s, \delta, \rho \rangle$$

конечный автомат, выдающий по любому входному суперслову периодическое суперслово

$$a^{p-1} b a^{p-1} b \dots a^{p-1} b \dots,$$

а через $f_p(x)$ – соответствующую одноместную автоматную функцию.

Покажем, что функцию $f_p(x)$ нельзя получить с помощью суперпозиции из функций f_1, \dots, f_m . Это следует из следующего замечания: если функция $f(x)$ является суперпозицией функций f_1, \dots, f_m , то она преобразует периодическую последовательность (a, \dots, a, \dots) периода 1 в периодическую последовательность с периодом T , все простые делители которого не превосходят N , а последовательность

$$a^{p-1} b a^{p-1} b \dots a^{p-1} b \dots$$

имеет простой период $p > N$. Полученное противоречие завершает доказательство теоремы. \square

В качестве примера применения введенных понятий рассмотрим теоретико-автоматную модель шифратора. Подобные модели позволяют анализировать современные симметрические шифры, реализуемые аппаратно – электронными схемами и программно – компьютерными программами. Существуют различные варианты теоретико-автоматных моделей шифраторов. Приведем одну из них, рассмотренную в лекциях Б.А. Погорелова. Напомним некоторые необходимые понятия, связанные с понятием шифра или шифрсистемы. Предварительно заметим, что шифрование в общем виде состоит в преобразовании открытых текстов в зашифрованные тексты. Пусть X – алфавит открытых текстов, а Y – алфавит зашифрованных текстов, а X^* и Y^* – соответствующие множества всех слов в этих алфавитах.

Пусть $\overline{X^*} \subseteq X^*$ и $\overline{Y^*} \subseteq Y^*$ – конечные подмножества соответственно возможных открытых и зашифрованных текстов.

Определение 7.5. *Шифром или шифрсистемой называется любая пятерка множеств вида*

$$\Sigma = \langle \overline{X^*}, K, \overline{Y^*}, E, D \rangle,$$

где

K – множество ключей,

$E = (E_k)_{k \in K}$ – семейство **правил зашифрования**: для каждого $k \in K$: E_k – отображение множества $\overline{X^*}$ открытых текстов во множество $\overline{Y^*}$ зашифрованных текстов,

$D = (D_k)_{k \in K}$ – семейство **правил расшифрования**: для каждого $k \in K$ D_k

– отображение множества $\overline{Y^*}$ шифрованных текстов во множество $\overline{X^*}$ открытых текстов.

При этом должны выполняться следующие условия:

1) для любого слова w из $\overline{X^*}$ и любого ключа k из K выполняется равенство

$$D_k(E_k(w)) = w,$$

$$2) \overline{Y^*} = \bigcup_{\substack{k \in K \\ w \in \overline{X^*}}} E_k(w).$$

Условие 1) означает, что рассматривается симметричная система шифрования – ключ расшифрования совпадает с ключом зашифрования. Кроме того из условия 1) следует, что для любого ключа k из K отображение $E_k : \overline{X^*} \rightarrow \overline{Y^*}$ является инъекцией.

Условие 2) носит во многом технический характер и несущественно для дальнейшего.

Принципиальное значение теоретико-автоматных методов для описания работы шифраторов состоит в возможности моделирования их функционирования в дискретном режиме.

Так как функции ρ_M и δ_M автомата M , рассматриваемого в качестве модели шифрсистемы, зависят от ключа k из множества ключей K , то с шифрсистемой естественно связать целое семейство $(M_k)_{k \in K}$ автоматов, которое может быть заменено одним автоматом:

если

$$M_k = \langle \Sigma, Q, \Omega, \delta_k, \rho_k \rangle,$$

то полагаем

$$M = \langle \Sigma, Q \times K, \Omega, \delta, \rho \rangle,$$

где

$$\delta((q_t, k), a_t) = (\delta_k(q_t, a_t), k) = (q_{t+1}, k), \quad \rho((q_t, k), a_t) = \rho_k(q_t, a_t) = b_{t+1}.$$

Иногда в модель шифрующего автомата включают две дополнительные функции

$\alpha : K \rightarrow Q$ – функция инициализации (выбора начального состояния),

$\beta : Q \times K \times \Sigma \rightarrow K$ – функция обновления ключа.

В этом случае, если выбрано начальное значение k_1 ключа, то в качестве начального состояния автомата выбирается $q_1 = \alpha(k_1)$.

Если рассматривается модель с функцией β обновления ключа, то к рассмотренным выше каноническим уравнениям автомата, задающим закон его функционирования во времени, добавляется уравнение смены ключа

$$k_{t+1} = \beta_M(q_t, k_t, a_t).$$

При этом если функция смены ключа не является тождественной, то шифрующий автомат называется мультиключевым.

Индукцией по длине $|w|$ слова w определяются функции

$$\delta_M^*(q, k, w), \quad \rho_M^*(q, k, w) \quad \text{и} \quad \beta_M^*(q, k, w)$$

и доказывается справедливость равенств

$$\begin{aligned} \delta_M^*(q, k, uw) &= \delta_M^*(\delta_M^*(q, k, u), \beta_M^*(q, k, u), w), \\ \rho_M^*(q, k, uw) &= \rho_M^*(q, k, u) \rho_M^*(\delta_M^*(q, k, u), \beta_M^*(q, k, u), w), \\ \beta_M^*(q, k, uw) &= \beta_M^*(\delta_M^*(q, k, u), \beta_M^*(q, k, u), w). \end{aligned}$$

С оценкой криптографической стойкости шифров тесно связана проблема минимизации для шифрующих автоматов.

Ключи k_1 и k_2 автомата M с функцией инициализации α называются *эквивалентными*, если совпадают реакции $M_{(\alpha(k_1), k_1)}$ и $M_{(\alpha(k_2), k_2)}$ автомата M , т.е. для любого слова w во входном алфавите Σ автомата M выполняется равенство

$$\rho_M^*(\alpha(k_1), k_1, w) = \rho_M^*(\alpha(k_2), k_2, w).$$

В качестве следствия теоремы Хаффмана – Мили получаем, что *для эквивалентности ключей k_1 и k_2 конечного шифрующего автомата*

$$M = \langle \Sigma, Q \times K, \Omega, \delta, \rho \rangle$$

необходимо и достаточно, чтобы выполнялось равенство

$$\rho_M^*(\alpha(k_1), k_1, w) = \rho_M^*(\alpha(k_2), k_2, w)$$

для любого слова w длины $|w| \leq |Q| \cdot |K| - 1$ во входном алфавите Σ автомата M . Поэтому *проблема эквивалентности ключей для конечного шифрующего автомата алгоритмически разрешима.*

Очевидно, что не каждый конечный автомат может рассматриваться в качестве модели шифратора. Это связано, в частности, с тем обстоятельством, что для любого ключа k из K отображение $E_k : \bar{X}^* \rightarrow \bar{Y}^*$ обратимо слева (является инъекцией). Поэтому мы приходим к необходимости рассмотрения вопроса об *обратимости слева* конечных автоматов.

Напомним понятие *автоматного отображения*. Если

$$M = \langle \Sigma, Q, \Omega, \delta, \rho \rangle$$

произвольный автомат, а q – его внутреннее состояние, то отображение M_q множества Σ^* всех слов во входном алфавите этого автомата во множество Ω^* всех слов в его выходном алфавите, заданное равенством

$$M_q(w) = \delta^*(q, w),$$

называется *автоматным отображением, соответствующим состоянию q , или реакцией состояния q .*

Определение 7.6. Автомат

$$\hat{M} = \langle \Omega, \hat{Q}, \Sigma, \hat{\delta}, \hat{\rho} \rangle$$

называется **левым обратным** к автомату

$$M = \langle \Sigma, Q, \Omega, \delta, \rho \rangle,$$

если для любого внутреннего состояния q автомата M найдется такое внутреннее состояние \hat{q} автомата \hat{M} , что для любого слова w во входном алфавите Σ автомата M выполняется равенство

$$\hat{M}_{\hat{q}}(M_q(w)) = w.$$

Аналогичным образом определяется понятие правого обратного автомата.

Определение 7.7. Автомат

$$\hat{M} = \langle \Omega, \hat{Q}, \Sigma, \hat{\delta}, \hat{\rho} \rangle$$

называется **правым обратным** к автомату

$$M = \langle \Sigma, Q, \Omega, \delta, \rho \rangle,$$

если для любого внутреннего состояния q автомата M найдется такое внутреннее состояние \hat{q} автомата \hat{M} , что для любого слова u во входном алфавите Ω автомата \hat{M} выполняется равенство

$$M_q(\hat{M}_{\hat{q}}(u)) = u.$$

Автомат M называется **инъективным**, если для любого его внутреннего состояния q инъективно автоматное отображение M_q .

Теорема 7.5. Для произвольного автомата M следующие три условия эквивалентны

- 1) для автомата M существует левый обратный автомат,
- 2) M – инъективный автомат,
- 3) для любого внутреннего состояния q инъективна функция $\rho_q(x) = \rho(q, x)$ из Σ в Ω .

Доказательство. Так как, очевидно, справедливы импликации $1) \implies 2) \implies 3)$, то для доказательства теоремы достаточно, например, доказать верность импликации $3) \implies 1)$.

Предположим, что для любого внутреннего состояния q инъективна функция $\rho_q(x) = \rho(q, x)$ из Σ в Ω . Тогда для любого символа b выходного алфавита Ω автомата M полный прообраз $\rho_q^{-1}(b)$ содержит не более одного элемента.

В случае, когда полный прообраз $\rho_q^{-1}(b)$ элемента b состоит из одного элемента a ($\rho(q, a) = b$), полагаем

$$\hat{\delta}(q, b) = \delta(q, a), \quad \hat{\rho}(q, b) = a.$$

Если же полный прообраз $\rho_q^{-1}(b)$ элемента b пуст, то задаем $\hat{\delta}(q, b)$ и $\hat{\rho}(q, b)$ произвольным образом.

Непосредственная проверка показывает, что автомат

$$\hat{M} = \langle \Omega, \hat{Q}, \Sigma, \hat{\delta}, \hat{\rho} \rangle$$

является левым обратным к автомату

$$M = \langle \Sigma, Q, \Omega, \delta, \rho \rangle.$$

□

Доказанная теорема обосновывает возможность понимать под шифрующим автоматом любой автомат вида

$$M = \langle \Sigma, Q \times K, \Omega, \delta, \rho \rangle,$$

где

$$\delta((q_t, k), a_t) = (\delta_k(q_t, a_t), k) = (q_{t+1}, k), \quad \rho((q_t, k), a_t) = \rho_k(q_t, a_t) = b_{t+1},$$

причем при любых фиксированных q и k функции $\rho((q, k), x)$ инъективны.

В заключение рассмотрим вопрос о различимости входов шифрующего автомата – его способности при любом ключе зашифровать два различных открытых текста начиная с некоторой длины в различные шифрованные тексты. Это приводит к следующему определению.

Определение 7.8. Слова (тексты) u и v во входном алфавите Σ шифрующего автомата

$$M = \langle \Sigma, Q \times K, \Omega, \delta, \rho, \alpha \rangle$$

с функцией инициализации α называются **неразличимыми на ключе k** , если для любого слова w во входном алфавите этого автомата выполняется равенство

$$\rho^*(\alpha(k), k, u \cdot w) = \rho^*(\alpha(k), k, v \cdot w).$$

Теорема 7.6. Слова u и v во входном алфавите Σ шифрующего автомата

$$M = \langle \Sigma, Q \times K, \Omega, \delta, \rho, \alpha \rangle$$

с функцией инициализации α неразличимыми на ключе k тогда и только тогда, когда выполняется равенство

$$\rho^*(\alpha(k), k, u) = \rho^*(\alpha(k), k, v)$$

и эквивалентны состояния $\delta^*(\alpha(k), k, u)$ и $\delta^*(\alpha(k), k, v)$ автомата M .

Д о к а з а т е л ь с т в о легко следует из равенств

$$\begin{aligned}\delta_M^*(q, k, uw) &= \delta_M^*(\delta_M^*(q, k, u), \beta_M^*(q, k, u), w), \\ \rho_M^*(q, k, uw) &= \rho_M^*(q, k, u) \rho_M^*(\delta_M^*(q, k, u), \beta_M^*(q, k, u), w), \\ \beta_M^*(q, k, uw) &= \beta_M^*(\delta_M^*(q, k, u), \beta_M^*(q, k, u), w).\end{aligned}$$

□

Следствие. Проблема распознавания различимости входов шифрующим автоматом алгоритмически разрешима.

ГЛАВА II

ЧАСТИЧНО РЕКУРСИВНЫЕ ФУНКЦИИ

§1. Примитивно рекурсивные и частично рекурсивные функции

Обозначим через N множество $\{0, 1, 2, \dots, \dots\}$ натуральных чисел. Заметим, что по чисто техническим причинам в это множество включен нуль 0. Можно было бы этого не делать, но тогда несколько усложнились бы некоторые доказательства.

Основным понятием для нас будет понятие *n -местной частичной числовой функции* f , которую мы будем называть просто *функцией*.

n -местной частичной числовой функцией f называется произвольное отображение некоторого подмножества $D(f)$ множества N^n во множество N . При этом множество $D(f)$ называется *областью определения* функции f .

n -местная функция f называется *всюду определенной*, если ее область определения $D(f)$ совпадает с множеством N^n .

Если нам по каким-либо причинам будет необходимо указать аргументы n -местной функции f , то мы часто будем использовать для этого выражение $f(x_1, \dots, x_n)$. При этом при $n = 1$ вместо $f(x_1)$ будем писать $f(x)$, при $n = 2$ вместо $f(x_1, x_2) = f(x, y)$, а при $n = 3$ вместо $f(x_1, x_2, x_3) = f(x, y, z)$.

Поясним одно из основных для нас понятий – понятие *вычислимой функции*.

n -местная частичная числовая функция f называется *вычислимой*, если существует алгоритм \mathcal{A} , который по произвольному набору $\langle a_1, \dots, a_n \rangle$ натуральных чисел, принадлежащему области определения $D(f)$ функции f , вычисляет значение $f(a_1, \dots, a_n)$ функции f на этом наборе.

Так как понятие "алгоритм \mathcal{A} ", о котором идет речь в предыдущем пояснении, понимается в интуитивном смысле (ему мы пока не дали точного математического определения), то и функция f называется *вычислимой в ин-*

интуитивном смысле. Однако мы будем говорить просто, что **функция f вычислима**, опуская слова "в интуитивном смысле".

Уточнить интуитивное понятие вычислимой функции, заменить его точным математическим понятием, в определенном смысле ему эквивалентным, можно, в принципе, двумя способами: либо *дав точное математическое определение фигурирующему в пояснении понятия вычислимой функции интуитивному понятию алгоритма*, либо *в точных математических терминах определив класс всех вычислимых функций*. В этом разделе будет рассмотрен второй подход – определение понятия вычислимой функции через понятие **частично рекурсивной функции**.

Простейшими или **исходными функциями** называются следующие функции:

нулевая функция – это одноместная функция, обозначаемая через $0(x)$ и при любом значении аргумента x принимающая в качестве значения число ноль 0,

функция следования – это одноместная функция, обозначаемая через $s(x)$ и при любом значении аргумента a принимающая в качестве значения $a + 1$, причем $a + 1$ – число, следующее за a , а не число, получаемое сложением a с единицей (операция сложения будет определена позже),

функции проектирования – это при любых $1 \leq t \leq n$ n -местная функция U_m^n , значение которой на произвольном наборе $\langle a_1, \dots, a_n \rangle$ натуральных чисел равно a_m , т.е. m -ой компоненте этого набора,

постоянные функции – это любая 0-местная функция, равная некоторой константе a и обозначаемая через a .

Таким образом в качестве исходных функций используется счетное множество весьма простых функций. Роль функций проектирования станет ясна несколько позже.

Для получения из исходных функций новых функций будут использоваться три специальных **оператора**.

Оператор суперпозиции. Если f_1, \dots, f_m – n -местные функции, g – m -местная функция, а n -местная функция f задается равенством

$$f(x_1, \dots, x_n) = g(f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)),$$

то будем говорить, что функция f получена из функций g, f_1, \dots, f_m применением **оператора суперпозиции** (с помощью оператора суперпозиции).

Будем использовать обозначение $f = S(g; f_1, \dots, f_m)$ для утверждения "функция f получена из функций g, f_1, \dots, f_m применением оператора суперпозиции".

Заметим, что оператор суперпозиции применим не к любым функциям – должны быть выполнены соответствующие требования (ограничения) на число аргументов. Однако по любому набору функций g, f_1, \dots, f_m легко определить, применим ли к ним оператор суперпозиции, т.е. выполнены ли соответствующие ограничения. При этом, если функции g, f_1, \dots, f_m вычислимы в интуитивном смысле либо всюду определены, то такова же и функция $S(g; f_1, \dots, f_m)$.

Близким к оператору суперпозиции является **оператор подстановки**.

Оператор подстановки. n -местная функция f получена из функций g, f_1, \dots, f_m применением **оператора подстановки** (с помощью оператора подстановки), если выполнено равенство

$$f(x_1, \dots, x_n) = g(t_1, \dots, t_m),$$

где каждое t_i либо удовлетворяет равенству вида $t_i = f_j(x_{j_1}, \dots, x_{j_k})$, где переменные x_{j_1}, \dots, x_{j_k} содержатся среди переменных x_1, \dots, x_n , либо t_i — одна из переменных x_1, \dots, x_n .

Следующая лемма часто будет использоваться явно и неявно в дальнейшем.

Лемма 1.1. Если n -местная функция f получена из функций g, f_1, \dots, f_m применением оператора подстановки, то она может быть получена из функций g, f_1, \dots, f_m и функций проектирования с помощью конечного числа применений оператора суперпозиции.

Д о к а з а т е л ь с т в о. сразу следует из следующего замечания: каждая переменная x_i может рассматриваться как n -местная функция проектирования $U_i^n(x_1, \dots, x_n)$. \square

Еще одним важным оператором для получения новых функций является **оператор примитивной рекурсии**.

Оператор примитивной рекурсии. Пусть g — n -местная функция, а h — $n+2$ -местная функции, последние два аргумента которой традиционно обозначаются через y и z . $n+1$ -местная функция f получается из функций g и h применением **оператора примитивной рекурсии**, если она удовлетворяет следующей системе равенств, которая называется **схемой примитивной рекурсии**

$$\begin{cases} f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n), \\ f(x_1, \dots, x_n, y+1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)). \end{cases}$$

Равенство $f = PR(g, h)$ будет служить сокращенной записью утверждения

”функция f получается из функций g и h применением оператора примитивной рекурсии”.

Заметим, что при $n = 0$ схема примитивной рекурсии принимает вид

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

где c — 0-местная функция, равная числу c .

Последним оператором будет **оператор минимизации**.

Оператор минимизации. Пусть g и h — $n+1$ -местные функции, последний аргумент которых традиционно обозначается через y . n -местная функция f получается из функций g и h применением **оператора минимизации**, если выполнено следующее условие:

для произвольных натуральных чисел a_1, \dots, a_n и b выполняется равенство $f(a_1, \dots, a_n) = b$ тогда и только тогда, когда при любом $t < b$ значения $g(a_1, \dots, a_n, t)$ и $h(a_1, \dots, a_n, t)$ определены и не равны, а значения $g(a_1, \dots, a_n, b)$ и $h(a_1, \dots, a_n, b)$ определены и равны.

Равенство $f(x_1, \dots, x_n) = \mu_y(g(x_1, \dots, x_n, y) = h(x_1, \dots, x_n, y))$ будет служить сокращенной записью утверждения

”функция f получается из функций g и h применением оператора минимизации”.

Оператор минимизации традиционно называют μ -оператором, чтобы отличить его от оператора, дающего наименьшее решение уравнения $g(x_1, \dots, x_n, y) = h(x_1, \dots, x_n, y)$. Например, наименьшее решение уравнения $y - 1 = x$ — это его единственное решение $x + 1$. Однако равенство $f(x) = \mu_y(y - 1 = x)$ задает нигде не определенную функцию.

Определим два важных класса функций — класс **примитивно рекурсивных функций** и класс **частично рекурсивных функций**.

Определение 1.1. Функция называется **примитивно рекурсивной**, если она может быть получена из исходных (простейших) функций посредством конечного числа применений операторов суперпозиции и примитивной рекурсии.

Заметим, что каждая примитивно рекурсивная функция является всюду определенной и вычислимой в интуитивном смысле. Это следует из следующих двух фактов:

1) все исходные функции являются всюду определенными и вычислимыми в интуитивном смысле;

2) операторы суперпозиции и примитивной рекурсии, будучи применены ко всюду определенным и вычислимым в интуитивном смысле функциям, дают всюду определенные и вычисляемые в интуитивном смысле функции.

Нашей ближайшей целью будет показать, насколько широк класс примитивно рекурсивных функций — он включает в себя многие известные теоретико-числовые функции. Из счетности класса примитивно рекурсивных функций и несчетности класса всех всюду определенных функций, конечно, следует существование всюду определенной функции, не являющейся примитивно рекурсивной. Однако привести пример всюду определенной вычислимой, но не примитивно рекурсивной функции не столь просто. Но такие примеры хорошо известны и будут указаны позже.

Приведем схемы примитивной рекурсии для последовательного доказательства примитивной рекурсивности важных теоретико-числовых функций.

Функция сложения $x + y$ задается следующей схемой примитивной рекурсии

$$\begin{cases} x_1 + 0 = x_1 = U_1^1(x_1), \\ x_1 + (y + 1) = (x_1 + y) + 1 = s(U_3^3(x_1, y, x_1 + y)). \end{cases}$$

Поэтому $+$ = $PR(U_1^1, S(s; U_3^3))$.

Функция умножения $x \cdot y$ задается следующей схемой примитивной рекурсии

$$\begin{cases} x_1 \cdot 0 = 0 = 0(U_1^1(x_1)), \\ x_1 \cdot (y + 1) = x_1 \cdot y + x_1 = S(+; U_3^3(x_1, y, x_1 \cdot y), U_1^3(x_1, y, x_1 \cdot y)). \end{cases}$$

Поэтому $\cdot = PR(S(0; U_1^1), S(+; U_3^3, U_1^3))$.

Функция $x \uparrow y = x^y$ задается следующей схемой примитивной рекурсии

$$\begin{cases} x_1^0 = 1, \\ x_1^{y+1} = x_1 \cdot x_1^y = S(\cdot; U_1^3(x_1, y, x_1^y), U_3^3(x_1, y, x_1^y)). \end{cases}$$

Поэтому $\uparrow = PR(1, S(\cdot; U_1^3, U_3^3))$.

Функция $x!$ задается следующей схемой примитивной рекурсии

$$\begin{cases} 0! = 1, \\ (x + 1)! = s(x) \cdot x! = S(\cdot; s(U_1^2(x, x!)), U_2^2(x, x!)). \end{cases}$$

Поэтому $! = PR(1, S(\cdot; S(s; U_1^2), U_2^2))$.

Индукцией по k для любого n определим постоянную n -местную функцию $k(x_1, \dots, x_n)$, равную тождественно k .

$$\begin{aligned} 0(x_1, \dots, x_n) &= 0(U_1^n(x_1, \dots, x_n)) \\ (k + 1)(x_1, \dots, x_n) &= s(k(x_1, \dots, x_n)). \end{aligned}$$

В дальнейшем важную роль играют следующие две функции sg и \overline{sg} – натуральные аналоги хорошо известной из математического анализа функции *sign* x – *сигнум* x – *знак числа* x .

$$sg\,x = \begin{cases} 0, & \text{если } x = 0, \\ 1, & \text{если } x > 0; \end{cases}$$

$$\overline{sg}\,x = \begin{cases} 1, & \text{если } x = 0, \\ 0, & \text{если } x > 0. \end{cases}$$

Эти функции задаются следующими схемами примитивной рекурсии:

$$\begin{cases} sg\,0 = 0, \\ sg\,(x + 1) = 1, \end{cases}$$

$$\begin{cases} \overline{sg}\,0 = 1, \\ \overline{sg}\,(x + 1) = 0. \end{cases}$$

Для получения по натуральному числу k предшествующего ему натурального числа служит функция

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

Эта функция задается следующей схемой примитивной рекурсии:

$$\begin{cases} \delta(0) = 0, \\ \delta(x + 1) = x. \end{cases}$$

Примитивно рекурсивным аналогом не всюду определенной функции $x - y$ служит функция $x \dot{-} y$ — *усеченная разность*

$$x \dot{-} y = \begin{cases} 0, & \text{если } x \leq y, \\ x - y, & \text{если } x > y. \end{cases}$$

Эта функция задается следующей схемой примитивной рекурсии:

$$\begin{cases} x \dot{-} 0 = x, \\ x \dot{-} (y + 1) = \delta(x \dot{-} y). \end{cases}$$

Из следующих равенств следует примитивная рекурсивность функций $|x - y|$, $\max(x, y)$ и $\min(x, y)$

$$\begin{aligned} |x - y| &= (x \dot{-} y) + (y \dot{-} x), \\ \max(x, y) &= x + (y \dot{-} x), \quad \min(x, y) = x \dot{-} (x \dot{-} y). \end{aligned}$$

Индукцией по n с использованием равенств

$$\max_{n+1}(x_1, \dots, x_n, x_{n+1}) = \max(\max_n(x_1, \dots, x_n), x_{n+1}),$$

$$\min_{n+1}(x_1, \dots, x_n, x_{n+1}) = \min(\min_n(x_1, \dots, x_n), x_{n+1})$$

устанавливается примитивная рекурсивность функций $\max_n(x_1, \dots, x_n)$ и $\min_n(x_1, \dots, x_n)$.

Определение 1.2. Функция называется **частично рекурсивной**, если она может быть получена из исходных (простейших) функций посредством конечного числа применений операторов суперпозиции, примитивной рекурсии и минимизации.

Заметим, что слова **частично рекурсивная функция** несколько не точно отражают смысл определяемого понятия – речь идет не о *частичной рекурсивности*, а о *частичной определенности*, не обязательно всюду определенности определяемых функций. Поэтому, возможно, более подходящим названием для этих функций было бы **рекурсивная частичная функция** или **частичная рекурсивная функция**. Такие предложения по изменению терминологии неоднократно высказывались, но мы будем использовать уже устоявшееся название **частично рекурсивная функция**.

Определим еще один класс функций – класс **рекурсивных функций**.

Определение 1.3. *Частично рекурсивная функция называется **рекурсивной**, если она является всюду определенной.*

Казалось бы, при поиске математического эквивалента интуитивного понятия вычислимой функции следовало бы ограничиться лишь всюду определенными функциями, но тогда даже такая простая функция, как разность $x - y$, не была бы вычислима, с чем трудно согласиться. О других причинах введения в рассмотрение не всюду определенных функций будет сказано позже.

Введем ряд операций, которые по примитивно рекурсивным (рекурсивным, частично рекурсивным) функциям дают примитивно рекурсивные (рекурсивные, частично рекурсивные) функции.

Теорема 1.1. *Если $n+1$ -местная функция f получена из $n+1$ -местной функции g посредством равенства*

$$f(x_1, \dots, x_n, y) = \sum_{i=0}^y g(x_1, \dots, x_n, i)$$

и g – примитивно рекурсивная (рекурсивная, частично рекурсивная) функция, то и f – примитивно рекурсивная (рекурсивная, частично рекурсивная) функция.

Про функцию f говорят, что она получена из функции g *суммированием*.

Д о к а з а т е л ь с т в о. Доказательство следует из равенств

$$\begin{aligned} f(x_1, \dots, x_n, 0) &= g(x_1, \dots, x_n, 0) = g(U_1^n(\bar{x}), \dots, U_n^n(\bar{x}), 0_1(U_1^n(\bar{x}))), \\ f(x_1, \dots, x_n, y+1) &= f(x_1, \dots, x_n, y) + g(x_1, \dots, x_n, y+1) = \\ &= U_{n+2}^{n+2}(\bar{x}, y, f(\bar{x}, y)) + \\ &= g(U_1^{n+2}(\bar{x}, y, f(\bar{x}, y)), \dots, U_n^{n+2}(\bar{x}, y, f(\bar{x}, y)), s(U_{n+1}^{n+2}(\bar{x}, y, f(\bar{x}, y)))). \end{aligned}$$

□

Следствие 1. *Если $n+2$ -местная функция f получена из $n+1$ -местной функции g посредством равенств*

$$f(x_1, \dots, x_n, y, z) = \begin{cases} \sum_{i=y}^z g(x_1, \dots, x_n, i), & \text{если } y \leq z, \\ 0, & \text{если } y > z \end{cases}$$

и g – примитивно рекурсивная (рекурсивная, частично рекурсивная) функция, то и f – примитивно рекурсивная (рекурсивная, частично рекурсивная) функция.

Доказательство следует из равенства

$$f(x_1, \dots, x_n, y, z) = \left(\sum_{i=0}^z g(x_1, \dots, x_n, i) - \sum_{i=0}^y g(x_1, \dots, x_n, i) \right) + g(x_1, \dots, x_n, y) \overline{sg}(y - z).$$

□

Следствие 2. Если n -местная функция f получена из n -местных функций g , α и β посредством равенства

$$f(x_1, \dots, x_n) = \sum_{i=\alpha(x_1, \dots, x_n)}^{\beta(x_1, \dots, x_n)} g(x_1, \dots, x_{n-1}, i)$$

и g , α и β – примитивно рекурсивные (рекурсивные, частично рекурсивные) функции, то и f – примитивно рекурсивная (рекурсивная, частично рекурсивная) функция.

Доказательство сразу следует из предыдущего следствия.

Аналогичные теоремы справедливы, если в их формулировках знак суммы \sum заменить знаком произведения \prod . Мы сформулируем пока лишь одну из таких теорем. Остальные будут вводиться по мере необходимости.

Теорема 1.2. Если $n+1$ -местная функция f получена из $n+1$ -местной функции g посредством равенства

$$f(x_1, \dots, x_n, y) = \prod_{i=0}^y g(x_1, \dots, x_n, i)$$

и g – примитивно рекурсивная (рекурсивная, частично рекурсивная) функция, то и f – примитивно рекурсивная (рекурсивная, частично рекурсивная) функция.

Про функцию f говорят, что она получена из функции g мультиплицированием.

Для дальнейшего нам потребуются примитивно рекурсивные и рекурсивные предикаты и отношения.

Напомним, что n -местный предикат P на множестве N – это любое отображение множества N^n во множество $\{, \}$, а n -местное отношение R на множестве N – это любое подмножество множества N^n . Заметим, что 1-местное отношение – это просто любое подмножество множества натуральных чисел N .

С каждым n -местным предикатом P на множестве N свяжем его *характеристическую функцию*

$$\chi_P(x_1, \dots, x_n) = \begin{cases} 1, & \text{если } P(x_1, \dots, x_n) \text{ истинно,} \\ 0, & \text{если } P(x_1, \dots, x_n) \text{ ложно.} \end{cases}$$

Аналогичным образом с каждым n -местным отношением R на множестве N свяжем его *характеристическую функцию*

$$\chi_R(x_1, \dots, x_n) = \begin{cases} 1, & \text{если } (x_1, \dots, x_n) \in R, \\ 0, & \text{если } (x_1, \dots, x_n) \notin R. \end{cases}$$

Определение 1.4. *Предикат или отношение называется примитивно рекурсивным или рекурсивным, если его характеристическая функция примитивно рекурсивна или рекурсивна.*

В частности, подмножество U множества натуральных чисел N называется примитивно рекурсивным или рекурсивным, если его характеристическая функция $\chi_U(x)$ примитивно рекурсивна или рекурсивна.

Покажем, что примитивно рекурсивны простейшие предикаты $=$, \neq , \leq и $<$. Это следует из равенств

$$\begin{aligned} \chi_=(x, y) &= \overline{sg}(|x - y|), & \chi_{\neq}(x, y) &= sg(|x - y|), \\ \chi_{\leq}(x, y) &= \overline{sg}(x - y), & \chi_{<}(x, y) &= sg(y - x). \end{aligned}$$

Теорема 1.3. *Классы примитивно рекурсивных и рекурсивных предикатов замкнуты относительно пропозициональных связок $\&$, \vee и \neg и относительно наложения ограниченных кванторов \forall_{\leq} , $\forall_{<}$, \exists_{\leq} и $\exists_{<}$.*

Доказательство. Для некоторого сокращения будем использовать \bar{x} в качестве обозначения набора x_1, \dots, x_n . Пусть $P(\bar{x})$ и $Q(\bar{x})$ — n -местные предикаты, а $\chi_P(\bar{x})$ и $\chi_Q(\bar{x})$ — их характеристические функции. Тогда доказательство первой части теоремы следует из равенств

$$\begin{aligned} \chi_{\neg P}(\bar{x}) &= \overline{sg}(\chi_P(\bar{x})), & \chi_{P\&Q}(\bar{x}) &= \chi_P(\bar{x}) \cdot \chi_Q(\bar{x}), \\ \chi_{P\vee Q}(\bar{x}) &= (\chi_P(\bar{x}) + \chi_Q(\bar{x})) \cdot \overline{sg}(\chi_P(\bar{x}) + \chi_Q(\bar{x})). \end{aligned}$$

Для доказательства второй части теоремы предположим, что $P(x_1, \dots, x_n)$ — n -местный предикат.

Обозначим через $((\forall)_{\leq} P)(x_1, \dots, x_n)$ n -местный предикат

$$(\forall y)_{y \leq x_n} P(x_1, \dots, x_{n-1}, y).$$

Тогда выполняется равенство

$$\chi_{((\forall)_{y \leq x_n} P)}(x_1, \dots, x_n) = \prod_{i=0}^{x_n} \chi_P(x_1, \dots, x_{n-1}, y).$$

Поэтому из примитивной рекурсивности или рекурсивности предиката $P(x_1, \dots, x_n)$ следует примитивная рекурсивность или рекурсивность предиката $(\forall y)_{y \leq x_n} P(x_1, \dots, x_{n-1}, y)$.

Для установления примитивной рекурсивности или рекурсивности предиката $(\forall y)_{y \leq x_n} P(x_1, \dots, x_{n-1}, y)$ достаточно заметить, что он равносильен предикату $(\forall y)_{y \leq x_n} (y = x_n \vee P(x_1, \dots, x_{n-1}, y))$.

Для установления примитивной рекурсивности или рекурсивности предиката $(\exists y)_{y \leq x_n} P(x_1, \dots, x_{n-1}, y)$ достаточно воспользоваться равенством

$$\chi_{((\exists)_{y \leq x_n} P)}(x_1, \dots, x_n) = \overline{sg}\left(\prod_{i=0}^{x_n} \overline{sg}(\chi_P(x_1, \dots, x_{n-1}, y))\right).$$

Для установления примитивной рекурсивности или рекурсивности предиката $(\exists y)_{y \leq x_n} P(x_1, \dots, x_{n-1}, y)$ достаточно заметить, что он равносильен предикату $(\exists y)_{y \leq x_n} (y \neq x_n \& P(x_1, \dots, x_{n-1}, y))$. \square

Рассмотрим один достаточно широко распространенный способ получения новых функций из имеющихся – задание функций кусочной схемой.

Пусть заданы m n -местных предикатов P_1, \dots, P_m таких, что ни на одном наборе натуральных чисел (a_1, \dots, a_n) не могут быть одновременно истинны никакие два из этих предикатов, т.е. тождественно истинен предикат

$$\bigwedge_{1 \leq i < j \leq m} \neg(P_i \& P_j).$$

Используя эти предикаты и произвольные $m+1$ n -местную функцию f_1, \dots, f_m и f_{m+1} определим новую n -местную функцию f равенствами

$$f(x_1, \dots, x_n) = \begin{cases} f_1(x_1, \dots, x_n), & \text{если } P_1(x_1, \dots, x_n) \text{ истинно,} \\ f_2(x_1, \dots, x_n), & \text{если } P_2(x_1, \dots, x_n) \text{ истинно,} \\ \dots & \dots, \\ f_m(x_1, \dots, x_n), & \text{если } P_m(x_1, \dots, x_n) \text{ истинно,} \\ f_{m+1}(x_1, \dots, x_n) & \text{в остальных случаях.} \end{cases}$$

Про функцию f говорят, что она задана кусочной схемой.

Теорема 1.4. Пусть n -местная функция f задана кусочной схемой

$$f(x_1, \dots, x_n) = \begin{cases} f_1(x_1, \dots, x_n), & \text{если } P_1(x_1, \dots, x_n) \text{ истинно,} \\ f_2(x_1, \dots, x_n), & \text{если } P_2(x_1, \dots, x_n) \text{ истинно,} \\ \dots & \dots, \\ f_m(x_1, \dots, x_n), & \text{если } P_m(x_1, \dots, x_n) \text{ истинно,} \\ f_{m+1}(x_1, \dots, x_n) & \text{в остальных случаях.} \end{cases}$$

Если функции f_1, \dots, f_m и f_{m+1} и предикаты P_1, \dots, P_m примитивно рекурсивны, рекурсивны либо частично рекурсивны, то такова же и функция f .

Доказательство. следует из равенства

$$f(\bar{x}) = f_1(\bar{x}) \cdot \chi_{P_1}(\bar{x}) + f_2(\bar{x}) \cdot \chi_{P_2}(\bar{x}) + \dots + f_m(\bar{x}) \cdot \chi_{P_m}(\bar{x}) + f_{m+1}(\bar{x}) \cdot \overline{sg}\left(\sum_{i=1}^m \chi_{P_m}(\bar{x})\right).$$

□

Если к примитивно рекурсивной функции применить μ -оператор, то даже если полученная функция окажется всюду определенной, она может не быть примитивно рекурсивной. Соответствующие примеры будут приведены позже.

Рассмотрим *ограниченный* μ -оператор. Предположим, что g — *всюду определенная* $n + 1$ -местная функция. Определим новую $n + 1$ -местную функцию $f(x_1, \dots, x_n, z)$, полагая

$$f(x_1, \dots, x_n, z) = \begin{cases} \mu_y (g(\bar{x}, y) = 0), & \text{если } (\exists y)_{y \leq z} (g(\bar{x}, y) = 0), \\ z + 1, & \text{в противном случае.} \end{cases}$$

Про функцию f говорят, что она получена применением *ограниченного* μ -оператора к функции g .

Теорема 1.5. Если функция f получена применением ограниченного μ -оператора к примитивно рекурсивной или рекурсивной функции g , то сама функция f примитивно рекурсивна или рекурсивна.

Доказательство. Утверждение теоремы следует из равенства

$$f(x_1, \dots, x_n, z) = \sum_{t=0}^z \prod_{i=0}^t sg(g(x_1, \dots, x_n, i)).$$

□

Следствие 1. Если функции $g(x_1, \dots, x_n, y)$ и $m(x_1, \dots, x_n)$ примитивно рекурсивны, причем

$$(\forall x_1) \dots (\forall x_n) (\exists y) (g(x_1, \dots, x_n, y) = 0 \ \& \ y \leq m(x_1, \dots, x_n))$$

и

$$f(x_1, \dots, x_n) = \mu_y (g(x_1, \dots, x_n, y) = 0),$$

то функция f примитивно рекурсивна.

Д о к а з а т е л ь с т в о. Пусть функция $h(x_1, \dots, x_n, z)$ получена из функции $g(x_1, \dots, x_n, y)$ применением ограниченного μ -оператора, тогда она примитивно рекурсивна. Для завершения доказательства остается заметить, что

$$f(x_1, \dots, x_n) = h(x_1, \dots, x_n, m(x_1, \dots, x_n)).$$

□

Пусть $P(x_1, \dots, x_n, y)$ — $n + 1$ -местный предикат. Тогда выражение

$$f(x_1, \dots, x_n) = \mu_y(P(x_1, \dots, x_n, y))$$

будет иметь тот же смысл, что и выражение

$$f(x_1, \dots, x_n) = \mu_y(\chi_P(x_1, \dots, x_n, y) = 1).$$

Установим примитивную рекурсивность некоторых теоретико-числовых функций.

Хорошо известно, что для любых натуральных чисел a и b существует единственная пара неотрицательных целых чисел q и r таких, что

$$a = bq + r \text{ и } 0 \leq r < b.$$

При этом q называется *неполным частным* от деления a на b и обозначается $[a/b]$ или $qt(a, b)$, а r называется *остатком* от деления a на b и обозначается $rest(a, b)$ или $rm(a, b)$.

Доопределим функции при $b = 0$, например, равенствами $qt(a, 0) = 0$ и $rest(a, 0) = a$, чтобы всегда было справедливо равенство

$$a = b \cdot qt(a, b) + rest(a, b).$$

Покажем, что функции $qt(x, y)$ и $rest(x, y)$ примитивно рекурсивны. Для этого достаточно воспользоваться равенствами

$$qt(x, y) = \mu_u((\exists z)_{z \leq x}(((y = 0 \& z = x \& u = 0) \vee (y > 0 \& z < x)) \& x = yu + z),$$

$$rest(x, y) = x - y \cdot qt(x, y).$$

Можно установить примитивную рекурсивность функций $qt(x, y)$ и $rest(x, y)$ и без использования оператора ограниченной минимизации. Однако рассуждение в этом случае несколько усложнится. Это сразу следует из равенств

$$rest(0, y) = 0,$$

$$rest(x + 1, y) = (rest(x, y) + 1)\chi_{<}(rest(x, y) + 1, y).$$

Определив функцию $h(x, y, z)$ равенством $h(x, y, z) = (z + 1)sg(|(z + 1) - y|)$, т.е. равенством $h(x, y, z) = s(z)sg(|s(z) - y|)$, мы зададим функцию $rest'(x, y) = rest(y, x)$ схемой примитивной рекурсии

$$rest'(x, 0) = 0(x),$$

$$rest'(x, y + 1) = h(x, y, rest'(x, y)).$$

И остается заметить, что

$$\text{rest}(x, y) = \text{rest}'(U_2^2(x, y), U_1^2(x, y)).$$

Рассмотрим отношение \mid делимости на множестве натуральных чисел. Так как $x \mid y$ тогда и только тогда, когда $(\exists z)_{z \leq y} y = x \cdot z$, то отношение \mid делимости является примитивно рекурсивным. Характеристическую функцию отношения \mid делимости обозначим через $\text{div}(x, y)$. Отметим, что $\text{div}(x, y) = \overline{\text{sg}}(\text{rest}(x, y))$. Из примитивной рекурсивности функции $\text{div}(x, y)$ следует примитивная рекурсивность функции $\text{nd}(x)$ – число делителей x , так как

$$\text{nd}(x) = \sum_{i=0}^x \text{div}(i, x).$$

Рассмотрим предикат $\text{Pr}(x)$ – “ x – простое число”. Так как характеристическая функция $\chi_{\text{Pr}}(x)$ предиката $\text{Pr}(x)$ – это $\overline{\text{sg}}(|\text{nd}(x) - 2|)$, то предикат $\text{Pr}(x)$ примитивно рекурсивен. Впрочем, это можно было бы установить и воспользовавшись эквивалентностью

$$\text{Pr}(x) \iff (x > 1) \& \neg((\exists y)_{y < x} (\exists z)_{z < x} x = y \cdot z).$$

В теории чисел важную роль играет функция $\pi(x)$, значение которой равно числу простых чисел, не превосходящих x . Ее примитивная рекурсивность следует из равенства

$$\pi(x) = \sum_{i=0}^x \chi_{\text{Pr}}(i).$$

Занумеруем натуральные числа в порядке возрастания

$$p_0 = 2, p_1 = 3, p_2 = 5, \dots, p_n, \dots,$$

где p_n – это n -ое простое число в этом пересчете. Построим примитивно рекурсивную функцию $p(n)$ такую, что при любом n $p(n) = p_n$.

Рассуждения, аналогичные доказательству теоремы Евклида о бесконечности множества простых чисел, показывают, что $p_{n+1} \leq (p_n)! + 1$. Поэтому

$$p(n+1) = (\mu_t)_{(t \leq p(n)!+1)} (p(n) < t \& \text{Pr}(t)).$$

Значит, задав функцию $h(y, z)$ равенством

$$h(y, z) = (\mu_t)_{(t \leq z!+1)} (z < t \& \text{Pr}(t)),$$

получим, что она примитивно рекурсивна. Тогда схема примитивной рекурсии

$$\begin{cases} p(0) = 2, \\ p(y+1) = h(y, p(y)) \end{cases}$$

задает функцию p . Поэтому множество простых чисел совпадает с множеством всех значений примитивно рекурсивной функции $p(n)$. Такие множества называются *рекурсивно перечислимыми*, но они будут рассмотрены более подробно позже.

В дальнейшем нам будет весьма полезна функция $ex(x, y)$ – экспонента простого числа p_x в числе y – это наибольший показатель степени, в которой простое число p_x делит число y .

По определению полагаем, что $ex(x, 0) = 0$. Кроме того заметим, что если a отлично от нуля и $p_x^u | a$, то $u \leq a$. Поэтому

$$ex(x, y + 1) = (\mu_t)_{t \leq y+1} (\neg(p(x)^{t+1} | (y + 1))).$$

Значит, функция $ex(x, y)$ примитивно рекурсивна.

Для произвольного отличного от нуля числа x обозначим через $lh(x)$ число ненулевых показателей в каноническом разложении числа x на простые множители. По определению полагаем $lh(0) = 0$. Для доказательства примитивной рекурсивности функции $lh(x)$ рассмотрим примитивно рекурсивный предикат

$$P(x, y) \iff (Pr(y) \& y | x).$$

Тогда примитивная рекурсивность функции $lh(x)$ следует из равенства

$$lh(x) = \sum_{i=0}^x \chi_P(x, i).$$

Для произвольного отличного от нуля числа x обозначим через $long(x)$ наибольшее натуральное число t такое, что простое число p_t делит x . По определению полагаем $long(0) = 0$. Для доказательства примитивной рекурсивности функции $long(x)$ воспользуемся при $x > 0$ равенством

$$long(x) = x - (\mu_t)_{(t \leq x)} (p(x - t) | x).$$

При арифметизации теории машин Тьюринга возникнет необходимость в нумерации слов в некотором алфавите, а для этого можно воспользоваться нумерацией Геделя конечных последовательностей положительных натуральных чисел. Геделевым номером последовательности $a = (a_0, a_1, \dots, a_k)$ положительных натуральных чисел называется число

$$n(a) = p_0^{a_0} p_1^{a_1} \dots p_k^{a_k}.$$

Назовем *конкатенацией*, или просто произведением последовательностей $a = (a_0, a_1, \dots, a_k)$ и $b = (b_0, b_1, \dots, b_m)$, последовательность

$$a \cdot b = (a_0, a_1, \dots, a_k, b_0, b_1, \dots, b_m).$$

Построим примитивно рекурсивную функцию $x * y$, которая по номерам двух последовательностей положительных натуральных чисел вычисляет номер их конкатенации, т.е.

$$n(a) * n(b) = p_0^{a_0} p_1^{a_1} \dots p_k^{a_k} \cdot p_{k+1}^{b_0} p_{k+2}^{b_1} \dots p_{k+m+1}^{b_m}.$$

Значит,

$$x * y = x \cdot \prod_{i=0}^{lh(y)} p(lh(x) + i)^{ex(i,y)}.$$

Пусть f – $n + 1$ -местная функция. Рассмотрим новую функцию

$$f^*(x_1, \dots, x_n, y) = \prod_{i=0}^y p_i^{f(x_1, \dots, x_n, i)}.$$

Лемма 2. Если функция f примитивно рекурсивна, рекурсивна или частично рекурсивна, то и функция f^* примитивно рекурсивна, рекурсивна или частично рекурсивна. Верно и обратное: если функция f^* примитивно рекурсивна, рекурсивна или частично рекурсивна, то и функция f примитивно рекурсивна, рекурсивна или частично рекурсивна.

Д о к а з а т е л ь с т в о следует из равенств

$$\begin{cases} f^*(x_1, \dots, x_n, 0) = 2^{f(x_1, \dots, x_n, 0)}, \\ f^*(x_1, \dots, x_n, y + 1) = f^*(x_1, \dots, x_n, y) \cdot p_{y+1}^{f(x_1, \dots, x_n, y+1)} \end{cases}$$

и

$$f(x_1, \dots, x_n, y) = ex(y, f^*(x_1, \dots, x_n, y)).$$

□

При рекурсивном определении тех или иных понятий, зависящих от некоторых параметров, в частности при рекурсивном задании функций, эти понятия вводятся для данных значений параметров через уже определенные для меньших значений параметров. Примитивная рекурсия – это простейший вид рекурсии. Рассмотрим несколько более общий вид рекурсии – возвратную рекурсию.

Предположим, что $s_1(t), \dots, s_k(t)$ – такие всюду определенные функции, что при любом t выполняются неравенства $s_j(t + 1) \leq t$ ($j = 1, \dots, k$). Пусть g – n -местная, h – $n + k + 1$ -местная функции. Схемой возвратной рекурсии называется система равенств вида

$$\begin{cases} f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n), \\ f(x_1, \dots, x_n, y + 1) = \\ \quad = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, s_1(y + 1)), \dots, f(x_1, \dots, x_n, s_k(y + 1))). \end{cases}$$

Теорема 1.6. Если функция f получается схемой возвратной рекурсии

$$\begin{cases} f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n), \\ f(x_1, \dots, x_n, y+1) = \\ \quad = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, s_1(y+1)), \dots, f(x_1, \dots, x_n, s_k(y+1))) \end{cases}$$

и функции $s_1(t), \dots, s_k(t)$, g и h — примитивно рекурсивны, рекурсивны или частично рекурсивны, то такова же и функция f .

Доказательство. Рассмотрим функцию

$$f^*(x_1, \dots, x_n, y) = \prod_{i=0}^y p_i^{f(x_1, \dots, x_n, i)}.$$

Так как $f(x_1, \dots, x_n, y) = ex(y, f^*(x_1, \dots, x_n, y))$, то достаточно доказать, что f^* может быть получена из функций $s_1(t), \dots, s_k(t)$, g , h и некоторых примитивно рекурсивных функций с помощью операторов суперпозиции и минимизации. Но это следует из равенств

$$f^*(x_1, \dots, x_n, 0) = 2^{g(x_1, \dots, x_n)},$$

$$\begin{aligned} f^*(x_1, \dots, x_n, y+1) &= f^*(x_1, \dots, x_n, y) \cdot p_{y+1}^{f(x_1, \dots, x_n, y+1)} = \\ &= f^*(x_1, \dots, x_n, y) \cdot p_{y+1}^{h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, s_1(y+1)), \dots, f(x_1, \dots, x_n, s_k(y+1)))} = \\ &= f^*(x_1, \dots, x_n, y) \cdot p_{y+1}^{h(x_1, \dots, x_n, y, ex(s_1(y+1), f^*(x_1, \dots, x_n, y)), \dots, ex(s_k(y+1), f^*(x_1, \dots, x_n, y)))}. \end{aligned}$$

□

§2. Нумерационные функции

В этом параграфе рассмотрим вопрос о нумерации наборов натуральных чисел. Начнем с простейшего случая — с нумерации пар. Так как множество пар натуральных чисел счетно, то все пары можно занумеровать натуральными числами. Сделать это можно, конечно, многими способами. Но нас будут интересовать лишь такие нумерации, для которых вопросы нахождения номера пары и восстановления компонент пары по ее номеру решаются с помощью примитивно рекурсивных функций. В качестве такой нумерации часто рассматривают канторовскую нумерацию: пары натуральных чисел располагаются в таблицу и нумеруются по диагоналям ”с северо-востока на юго-запад”. В итоге получаем

$$\begin{aligned} (0, 0), (0, 1), (1, 0), (0, 2), (1, 1), (2, 0), \dots \\ (0, x+y), (1, x+y-1), \dots, (x, y), \dots, (x+y-1, x), (x+y, 0), \dots \end{aligned}$$

Номер пары (a, b) при такой нумерации называется канторовским номером и обозначается через $c(a, b)$.

Нетрудно проверить, что

$$c(x, y) = (x + y)(x + y + 1)/2 + x = ((x + y)^2 + 3x + y)/2 = [(x + y)^2 + 3x + y]/2.$$

Поэтому $c(x, y)$ – примитивно рекурсивная функция.

Так как $x \leq c(x, y)$ и $y \leq c(x, y)$, то можно весьма просто определить примитивно рекурсивные функции $l(z)$ и $r(z)$ такие, что

$$l(c(x, y)) = x, \quad r(c(x, y)) = y, \quad c(l(z), r(z)) = z.$$

Для этого полагаем

$$l(z) = (\mu_x)_{x \leq z} (\exists y)_{y \leq z} (2z = (x + y)^2 + 3x + y), \\ r(z) = (\mu_y)_{y \leq z} (\exists x)_{x \leq z} (2z = (x + y)^2 + 3x + y).$$

Тройка функций c , l и r носит название *нумерационной системы функций для пар натуральных чисел*. Конечно, существует и много других троек нумерационных функций C , L и R , для которых выполнены равенства

$$L(C(x, y)) = x, \quad R(C(x, y)) = y, \quad C(L(z), R(z)) = z.$$

Индукцией по n построим нумерационные функцию c_n , $c_n^{(1)}$, \dots , $c_n^{(n)}$ для n -ок натуральных чисел, удовлетворяющие равенствам

$$c_n(c_n^{(1)}(z), \dots, c_n^{(n)}(z)) = z, \\ c_n^{(1)}(c(x_1, \dots, x_n)) = x_1, \dots, c_n^{(n)}(c(x_1, \dots, x_n)) = x_n.$$

Полагаем

$$c_2^{(1)} = l, \quad c_2^{(2)} = r, \\ c_{n+1}(x_1, \dots, x_n, x_{n+1}) = c(c_n(x_1, \dots, x_n), x_{n+1}),$$

$$c_{n+1}^{(n+1)}(z) = r(z),$$

$$c_{n+1}^{(n)}(z) = rl(z),$$

$$c_{n+1}^{(n-1)}(z) = rll(z),$$

$$\dots\dots\dots,$$

$$c_{n+1}^{(2)}(z) = rll \dots l(z),$$

$$c_{n+1}^{(1)}(z) = lll \dots l(z).$$

Построенные нумерационные функции позволяют сводить изучение n -местных функций $f(x_1, \dots, x_n)$ к изучению одноместных функций

$$g(x) = f(c_n^{(1)}(x), \dots, c_n^{(n)}(x)),$$

так как

$$f(x_1, \dots, x_n) = g(c_n(x_1, \dots, x_n)).$$

Построенные нумерационные функции при фиксированном n осуществляют взаимно однозначную нумерацию n -ок натуральных чисел, т.е. у каждого набора в точности один номер. При нумерации других объектов от такого требования часто можно отказаться, достаточно лишь потребовать, чтобы у каждого из нумеруемых объектов был номер, не обязательно единственный, но вычисляемый по объекту. Желательно существование алгоритма, проверяющего по произвольному натуральному числу, является ли оно номером некоторого из нумеруемых объектов и в случае положительного ответа восстанавливающего соответствующий объект. Воздержимся от дальнейших уточнений.

В качестве важного для нас примера рассмотрим *геделеву нумерацию всех конечных последовательностей натуральных чисел*. Будет построена *примитивно рекурсивная функция* $\Gamma(x, y)$ такая, что для любой конечной последовательности натуральных чисел a_0, a_1, \dots, a_n (произвольной длины n) найдется такое число m , что

$$\bigwedge_{i=0}^n \Gamma(m, i) = a_i.$$

Построение функции $\Gamma(x, y)$ можно выполнить, например, при помощи *китайской теоремы об остатках*.

Рассмотрим функцию от трех переменных $rest(u, 1 + (y + 1)z)$. Для произвольного набора натуральных чисел a_0, a_1, \dots, a_n найдем такое число b , чтобы были взаимно просты числа $m_y = 1 + (y + 1)b$ при $y = 0, 1, \dots, n$. Достаточно в качестве b взять $(1 + n + a_0 + a_1 + \dots + a_n)!$. Тогда по китайской теореме об остатках система сравнений

$$\bigwedge_{i=0}^n x \equiv a_i \pmod{m_i}$$

имеет натуральное решение A . А так как $a_i < m_i$, то $rest(A, m_i) = a_i$.

Поэтому в качестве функции $\Gamma(x, y)$ можно взять

$$rest(l(x), 1 + (y + 1)r(x)).$$

§3. Рекурсивно перечислимые множества и предикаты

В этом параграфе будет обсуждаться одно из важнейших понятий теории алгоритмов – понятие **рекурсивно перечислимого множества и предиката**. Из различных равносильных определений этого понятия мы выбираем то, которое, на наш взгляд, "сразу" как-то объясняет слово "перечислимое".

Определение 3.1. *Непустое множество натуральных чисел называется рекурсивно перечислимым, если оно совпадает с множеством всех значений*

некоторой примитивно рекурсивной функции. Пустое множество по определению считается также рекурсивно перечислимым.

Теорема 3.1. *Множество U натуральных чисел рекурсивно перечисливо тогда и только тогда, когда найдется такая 2-местная примитивно рекурсивная функция $F(x, y)$, что для произвольного натурального числа a справедлива эквивалентность*

$$a \in U \iff (\exists y)F(a, y) = 0.$$

Д о к а з а т е л ь с т в о. Пусть U – непустое рекурсивно перечислимое множество. Найдется такая одноместная примитивно рекурсивная функция f , что

$$U = \{f(t) \mid t \in N\}.$$

Тогда

$$a \in U \iff (\exists y)|a - f(y)| = 0.$$

И в качестве примитивно рекурсивной функции $F(x, y)$ можно взять $|x - f(y)|$.

Для доказательства обратного утверждения предположим, что для множества U и примитивно рекурсивной функции $F(x, y)$ для произвольного натурального числа a справедлива эквивалентность

$$a \in U \iff (\exists y)F(a, y) = 0.$$

Если множество U пусто, то оно рекурсивно перечисливо по определению.

Пусть U – непустое множество. Возьмем $a \in U$. Обозначим через $f(t)$ функцию

$$l(t) \cdot \overline{sg}(F(l(t), r(t))) + a \cdot sg(F(l(t), r(t))).$$

Нетрудно понять, что

$$U = \{f(t) \mid t \in N\}.$$

□

Класс рекурсивно перечислимых множеств замкнут относительно операций объединения и пересечения. С дополнением ситуация более сложная.

Теорема 3.2. *Объединение и пересечение двух рекурсивно перечислимых множеств рекурсивно перечислимы.*

Д о к а з а т е л ь с т в о. Пусть для множеств U_i ($i = 1, 2$) и примитивно рекурсивных функций $F(x, y)_i$ для произвольного натурального числа a справедливы эквивалентности

$$a \in U_i \iff (\exists y)F_i(a, y) = 0.$$

Тогда

$$\begin{aligned} a \in (U_1 \cup U_2) &\iff (\exists y)F_1(a, y) \cdot F_2(a, y) = 0, \\ a \in (U_1 \cap U_2) &\iff (\exists y)F_1(a, l(y)) + F_2(a, r(y)) = 0. \end{aligned}$$

□

Теорема 3.3 (Э. Пост). *Если множество U и его дополнение \bar{U} рекурсивно перечислимые множества, то U – рекурсивное множество.*

До к а з а т е л ь с т в о. Пусть для множества U и примитивно рекурсивных функций $F(x, y)$ и $H(x, y)$ для произвольного натурального числа a справедливы эквивалентности

$$a \in U \iff (\exists y)F(a, y) = 0,$$

$$a \notin U \iff (\exists y)H(a, y) = 0.$$

Рассмотрим функцию

$$f(x) = \mu_y (F(x, y) \cdot H(x, y) = 0).$$

Тогда f – рекурсивная функция и

$$\chi_U(x) = \overline{sg}(F(x, f(x))).$$

Поэтому U – рекурсивное множество. \square

Позже будет показано, что верна и обратная теорема, т.е. *любое рекурсивное множество является рекурсивно перечислимым*. Но для ее доказательства нам необходимо более подробно познакомиться с рекурсивно перечислимыми множествами.

Рассмотрим понятие рекурсивной перечислимости для n -ок натуральных чисел. Пусть A – непустое подмножество множества N^n . Возможны, например, следующие два определения рекурсивной перечислимости множества A .

Определение (I). *Непустое множество A n -ок натуральных чисел называется рекурсивно перечислимым, если существуют такие примитивно рекурсивные функции $\alpha_1(t), \dots, \alpha_n(t)$, что*

$$A = \{ (\alpha_1(t), \dots, \alpha_n(t)) \mid t \in N \}.$$

Пустое множество рекурсивно перечислимо.

Теорема 3.4. *Непустое множество A n -ок натуральных чисел рекурсивно перечислимо (в смысле определения I) тогда и только тогда, когда рекурсивно перечислимо множество*

$$c(A) = \{ c_n(a_1, \dots, a_n) \mid (a_1, \dots, a_n) \in A \}$$

номеров его элементов.

До к а з а т е л ь с т в о. Если

$$A = \{ (\alpha_1(t), \dots, \alpha_n(t)) \mid t \in N \},$$

то

$$c(A) = \{ c_n(\alpha_1(t), \dots, \alpha_n(t)) \mid t \in N \}.$$

Обратно, если

$$c(A) = \{ f(t) \mid t \in N \},$$

то

$$A = \{ (c_n^{(1)}(f(t)), \dots, c_n^{(n)}(f(t)) \mid t \in N \}.$$

□

Определение (II). Множество A n -ок натуральных чисел называется рекурсивно перечислимым, если существует такая примитивно рекурсивная функция $f(x_1, \dots, x_n, y)$, что

$$(a_1, \dots, a_n) \in A \iff (\exists y)f(a_1, \dots, a_n, y) = 0.$$

Теорема 3.5. Множество A n -ок натуральных чисел рекурсивно перечислимо (в смысле определения II) тогда и только тогда, когда рекурсивно перечислимо множество $c(A)$ номеров его элементов.

Д о к а з а т е л ь с т в о. Если

$$(a_1, \dots, a_n, y) \in A \iff (\exists y)f(a_1, \dots, a_n, y) = 0,$$

то

$$m \in A \iff (\exists y)f(c_n^{(1)}(m), \dots, c_n^{(n)}(m), y) = 0.$$

Обратно, если

$$m \in c(A) \iff (\exists y)f(m, y) = 0,$$

то

$$(a_1, \dots, a_n, y) \in A \iff (\exists y)f(c_n(a_1, \dots, a_n), y) = 0.$$

□

Следствие. Определения I и II равносильны.

Из выше доказанных теорем следует, что объединение и пересечение двух рекурсивно перечислимых множеств n -ок натуральных чисел рекурсивно перечислимы.

Кроме того, сделаем следующее полезное в дальнейшем замечание.

Предположим, что для множества A n -ок натуральных чисел существует такая примитивно рекурсивная функция $f(x_1, \dots, x_n, y_1, \dots, y_m)$, что

$$(a_1, \dots, a_n) \in A \iff (\exists y_1) \dots (\exists y_m) f(a_1, \dots, a_n, y_1, \dots, y_m) = 0.$$

Покажем, что множество A рекурсивно перечислимо. Это следует из эквивалентности

$$(a_1, \dots, a_n) \in A \iff (\exists t)f(a_1, \dots, a_n, c_m^{(1)}(t), \dots, c_m^{(m)}(t)) = 0.$$

Для множеств n -ок натуральных чисел определены еще две специфические операции – ”проектирование, или навешивание квантора существования” и ”навешивание ограниченного квантора общности”, не выводящие за пределы класса рекурсивно перечислимых множеств.

Теорема 3.6. *Если A – рекурсивно перечислимое множество n -ок натуральных чисел, то следующие два множества рекурсивно перечислимы*

$$\{(a_1, \dots, a_{n-1}) \mid (\exists y)(a_1, \dots, a_{n-1}, y) \in A\},$$

$$\{(a_1, \dots, a_{n-1}, a_n) \mid (\forall y)_{y \leq a_n} (a_1, \dots, a_{n-1}, y) \in A\}.$$

Д о к а з а т е л ь с т в о. Введем обозначения

$$pr(A) = \{(a_1, \dots, a_{n-1}) \mid (\exists y)(a_1, \dots, a_{n-1}, y) \in A\},$$

$$\forall(A) = \{(a_1, \dots, a_{n-1}, a_n) \mid (\forall y)_{y \leq a_n} (a_1, \dots, a_{n-1}, y) \in A\}.$$

Если множество A n -ок натуральных чисел рекурсивно перечислимо, то существует такая примитивно рекурсивная функция $f(x_1, \dots, x_n, y)$, что

$$(a_1, \dots, a_n) \in A \iff (\exists y)f(a_1, \dots, a_n, y) = 0.$$

Тогда

$$(a_1, \dots, a_{n-1}) \in pr(A) \iff (\exists y)f(a_1, \dots, a_{n-1}, l(y), r(y)) = 0$$

и

$$(a_1, \dots, a_n) \in \forall(A) \iff (\exists y) \sum_{t=0}^{a_n} f(a_1, \dots, a_{n-1}, t, \Gamma(y, t)) = 0,$$

где $\Gamma(y, t)$ – функция Геделя. □

Напомним, что *графиком* n -местной функции $f(x_1, \dots, x_n)$ называется множество

$$\Gamma_f = \{(a_1, \dots, a_n, f(a_1, \dots, a_n)) \mid (a_1, \dots, a_n) \in D(f)\}.$$

Имеет место следующая достаточно важная теорема, которая существенно расширяет круг наших знаний о рекурсивно перечислимых множествах.

Теорема 3.7 (О графике частично рекурсивной функции). *Для того чтобы функция f была частично рекурсивной необходимо, и достаточно, чтобы ее график был рекурсивно перечислимым множеством.*

Д о к а з а т е л ь с т в о. Если функция f нигде не определена, то ее график Γ_f пуст, а значит, утверждение теоремы для нигде не определенных функций тривиально справедливо.

Пусть f – функция с непустой областью определения $D(f)$, а значит, и с непустым графиком Γ_f .

Если график Γ_f функции f рекурсивно перечислим, то найдутся такие примитивно рекурсивные функции $\alpha_1(t), \dots, \alpha_n(t)$ и $\beta(t)$, что

$$\Gamma_f = \{ (\alpha_1(t), \dots, \alpha_n(t), \beta(t)) \mid t \in N \}.$$

Рассмотрим функцию

$$g(x_1, \dots, x_n) = \mu_t \left(\bigwedge_{i=1}^n x_i = \alpha_i(t) \right).$$

Тогда

$$f(x_1, \dots, x_n) = \beta(g(x_1, \dots, x_n)).$$

Поэтому *функция f частично рекурсивна*. Отметим еще одно важное обстоятельство: *для получения функции f μ -оператор используется лишь один раз*.

Для доказательства обратного утверждения *график любой частично рекурсивной функции рекурсивно перечислим* применим уже использовавшуюся неоднократно выше схему:

- 1) докажем рекурсивную перечислимость графиков простейших функций,
- 2) установим замкнутость класса функций с рекурсивно перечислимыми графиками относительно операторов суперпозиции, примитивной рекурсии и минимизации.

Справедливость первого утверждения следует из следующих соотношений:

$$\begin{aligned} \Gamma_{0_1} &= \{ (U_1^{(1)}(x_1), 0_1(U_1^{(1)}(x_1))) \mid x_1 \in N \}, \\ \Gamma_s &= \{ (U_1^{(1)}(x_1), s(U_1^{(1)}(x_1))) \mid x_1 \in N \}, \\ \Gamma_{U_n^{(k)}} &= \{ (c_n^{(1)}(t), \dots, c_n^{(n)}(t), c_n^{(k)}(t)) \mid t \in N \}. \end{aligned}$$

Докажем замкнутость класса функций с рекурсивно перечислимыми графиками относительно операторов суперпозиции, примитивной рекурсии и минимизации.

Пусть функция $f = S(g; f_1, \dots, f_m)$ получена применением оператора суперпозиции к функциям g, f_1, \dots и f_m с рекурсивно перечислимыми графиками $\Gamma_g, \Gamma_{f_1}, \dots$ и Γ_{f_m} .

Покажем, что график Γ_f функции f рекурсивно перечислим.

Если хотя бы один из графиков $\Gamma_g, \Gamma_{f_1}, \dots$ и Γ_{f_m} пуст, то пуст и график Γ_f функции f . Поэтому он рекурсивно перечислим по определению.

Пусть

$$\begin{aligned} \Gamma_{f_i} &= \{ (\alpha_{i,1}(t), \dots, \alpha_{i,n}(t), \alpha_i(t)) \mid t \in N \}, \\ \Gamma_g &= \{ (\beta_1(t), \dots, \beta_m(t), \beta(t)) \mid t \in N \}. \end{aligned}$$

Заметим, что

$$\begin{aligned} (x_1, \dots, x_n, y) \in \Gamma_f &\iff \\ &(\exists z_1) \dots (\exists z_m) ((x_1, \dots, x_n, z_1) \in \Gamma_{f_1} \& \dots \& \\ &(x_1, \dots, x_n, z_m) \in \Gamma_{f_m} \& (z_1, \dots, z_m, y) \in \Gamma_g). \end{aligned}$$

Это дает эквивалентность

$$\begin{aligned}
 (x_1, \dots, x_n, y) \in \Gamma_f &\iff (\exists t_0)(\exists t_1) \dots (\exists t_m) \\
 &\quad (x_1 = \alpha_{1,1}(t_1) \quad \& \dots \& x_n = \alpha_{1,n}(t_1) \& \\
 &\quad x_1 = \alpha_{2,1}(t_2) \quad \& \dots \& x_n = \alpha_{2,n}(t_2) \& \\
 &\quad \dots \dots \dots \dots \dots \dots \dots \\
 &\quad x_1 = \alpha_{m,1}(t_m) \quad \& \dots \& x_n = \alpha_{m,n}(t_m) \& \\
 &\quad \beta_1(t_0) = \alpha_1(t_1) \quad \& \dots \& \beta_m(t_0) = \alpha_m(t_m) \& \\
 &\quad \beta(t_0) = y).
 \end{aligned}$$

Пусть функция $f = PR(g; h)$ получена применением оператора примитивной рекурсии к функциям g и h с рекурсивно перечислимыми графиками Γ_g и Γ_h .

Покажем, что график Γ_f функции f рекурсивно перечислим.

Если хотя бы один из графиков Γ_g или Γ_h пуст, то пуст и график Γ_f функции f . Поэтому он рекурсивно перечислим.

Пусть

$$\Gamma_h = \{ (\alpha_1(t), \dots, \alpha_n(t), \alpha_{n+1}(t), \alpha_{n+2}(t), \alpha(t)) \mid t \in N \},$$

$$\Gamma_g = \{ (\beta_1(t), \dots, \beta_n(t), \beta(t)) \mid t \in N \}.$$

Напомним, что $f(x_1, \dots, x_n, y) = z$ тогда и только тогда, когда существует последовательность натуральных чисел b_0, b_1, \dots, b_y такая, что

$$b_0 = g(x_1, \dots, x_n) \& (\forall t)_{t < y} b_{t+1} = h(x_1, \dots, x_n, t, b_t) \& b_y = z.$$

Последнее равносильно существованию последовательности натуральных чисел t_0, t_1, \dots, t_y такой, что

$$\begin{aligned}
 (x_1 = \beta_1(t_0) \& \dots \& x_n = \beta_n(t_0) \& b_0 = \beta(t_0) \& \beta(t_y) = z \& \\
 &\quad (x_1 = \alpha_1(t_1) \quad \& \dots \& x_n = \alpha_n(t_1) \quad \& \\
 &\quad 0 = \alpha_{n+1}(t_1) \quad \& b_0 = \alpha_{n+2}(t_1) \quad \& b_1 = \alpha(t_1) \quad \& \\
 &\quad x_1 = \alpha_1(t_2) \quad \& \dots \& x_n = \alpha_n(t_2) \quad \& \\
 &\quad 1 = \alpha_{n+1}(t_2) \quad \& b_1 = \alpha_{n+2}(t_2) \quad \& b_2 = \alpha(t_2) \quad \& \\
 &\quad \dots \dots \dots \dots \dots \dots \dots \\
 &\quad x_1 = \alpha_1(t_y) \quad \& \dots \& x_n = \alpha_n(t_y) \quad \& \\
 &\quad y - 1 = \alpha_{n+1}(t_y) \quad \& b_{y-1} = \alpha_{n+2}(t_y) \quad \& b_y = \alpha(t_y) \quad).
 \end{aligned}$$

Так как последовательность натуральных чисел t_0, t_1, \dots, t_y может иметь произвольную длину, то воспользуемся функцией Геделя $\Gamma(x, y)$: найдем такое

u , что при любом $0 \leq s \leq y$ $\Gamma(u, s) = t_s$. Получаем эквивалентность

$$\begin{aligned} (x_1, \dots, x_n, y, z) \in \Gamma_f &\iff \\ &(\exists u)((x_1 = \beta_1(\Gamma(u, 0)) \& \dots \& x_n = \beta_n(\Gamma(u, 0))) \& \\ &((y = 0 \& z = \beta(\Gamma(u, 0)) \vee \\ &((y > 0) \& (\forall t)_{1 \leq t < y} (x_1 = \alpha_1(\Gamma(u, t)) \& \dots \& x_n = \alpha_n(\Gamma(u, t))) \& \\ &t = \alpha_{n+1}(\Gamma(u, t))) \& \& \alpha(\Gamma(u, t)) = \alpha_{n+2}(\Gamma(u, t+1)) \& \\ &\beta(\Gamma(u, 0)) = \alpha_{n+2}(\Gamma(u, 1)) \& z = \alpha(\Gamma(u, y))). \end{aligned}$$

Из этой эквивалентности следует, что график Γ_f функции f рекурсивно перечислим.

Пусть функция $f(x_1, \dots, x_n) = \mu_y(g(x_1, \dots, x_n, y) = 0)$ получена применением μ -оператора к функции g с рекурсивно перечислимым графиком Γ_g .

Покажем, что график Γ_f функции f рекурсивно перечислим.

Если график Γ_g пуст, то пуст и график Γ_f функции f . Поэтому он рекурсивно перечислим.

Пусть

$$\Gamma_g = \{ (\alpha_1(t), \dots, \alpha_n(t), \alpha_{n+1}(t), \alpha(t)) \mid t \in N \}.$$

Напомним, что $f(x_1, \dots, x_n) = y$ тогда и только тогда, когда существует последовательность натуральных чисел b_0, b_1, \dots, b_y такая, что

$$(\forall t)_{t \leq y} (b_t = g(x_1, \dots, x_n, t)) \& (\forall t)_{t < y} b_t \neq 0 \& b_y = 0.$$

Последнее равносильно существованию последовательности натуральных чисел t_0, t_1, \dots, t_y такой, что

$$\begin{array}{lll} (x_1 = \alpha_1(t_0)) & \& \dots \& x_n & = \alpha_n(t_0) \& \\ 0 = \alpha_{n+1}(t_0) & \& b_0 = \alpha(t_0) & & \& b_0 \neq 0 \& \\ x_1 = \alpha_1(t_1) & \& \dots \& x_n & = \alpha_n(t_1) \& \\ 1 = \alpha_{n+1}(t_1) & \& b_1 = \alpha(t_1) & & \& b_1 \neq 0 \& \\ \dots\dots\dots & \dots\dots\dots & \dots\dots\dots & & \dots\dots\dots \\ x_1 = \alpha_1(t_{y-1}) & \& \dots \& x_n & = \alpha_n(t_{y-1}) \& \\ y-1 = \alpha_{n+1}(t_{y-1}) & \& b_{y-1} = \alpha(t_{y-1}) & & \& b_{y-1} \neq 0 \& \\ x_1 = \alpha_1(t_y) & \& \dots \& x_n & = \alpha_n(t_y) \& \\ y = \alpha_{n+1}(t_y) & \& b_y = \alpha(t_y) & & \& b_y = 0 \& \end{array}$$

Так как последовательность натуральных чисел t_0, t_1, \dots, t_y может иметь произвольную длину, то вновь воспользуемся функцией Геделя $\Gamma(x, y)$: найдем

такое u , что при любом $0 \leq s \leq y$ $\Gamma(u, s) = t_s$. Получаем эквивалентность

$$(x_1, \dots, x_n, y, z) \in \Gamma_f \iff (\exists u)((\forall t)_{t \leq y}((x_1 = \alpha_1(\Gamma(u, t)) \& \dots \& x_n = \alpha_n(\Gamma(u, t)) \& t = \alpha_{n+1}(\Gamma(u, t))) \& (\forall t)_{t < y}(\alpha(\Gamma(u, t)) \neq 0) \& \alpha(\Gamma(u, y)) = 0).$$

Из этой эквивалентности следует, что график Γ_f функции f рекурсивно перечислим. \square

В качестве следствия доказанной теоремы получаем следующую теорему.

Теорема 3.8. *Область определения $D(f)$ и множество значений $R(f)$ любой частично рекурсивной функции f являются рекурсивно перечислимыми множествами.*

Д о к а з а т е л ь с т в о. Если частично рекурсивная функция f нигде не определена, то ее область определения $D(f)$ и множество значений $R(f)$ пусты. Значит они рекурсивно перечислимы.

Пусть область определения $D(f)$ частично рекурсивной функции f – непустое множество. Тогда график Γ_f этой функции f имеет вид

$$\Gamma_f = \{(\alpha_1(t), \dots, \alpha_n(t), \alpha(t)) \mid t \in N\},$$

где $\alpha_1(t), \dots, \alpha_n(t)$ и $\alpha(t)$ – некоторые примитивно рекурсивные функции. Тогда

$$D(f) = \{(\alpha_1(t), \dots, \alpha_n(t)) \mid t \in N\}$$

и

$$R(f) = \{\alpha(t) \mid t \in N\}.$$

Поэтому область определения $D(f)$ и множество значений $R(f)$ любой частично рекурсивной функции f являются рекурсивно перечислимыми множествами. \square

Верно и обратное утверждение: каждое рекурсивно перечислимое множество есть область определения некоторой частично рекурсивной функции. Напомним, что по определению любое рекурсивно перечислимое множество есть множество значений некоторой примитивно рекурсивной функции.

Рассмотрим частичную характеристическую функцию множества n – ок U натуральных чисел

$$\chi_U^{(p)}(x_1, \dots, x_n) = \begin{cases} 1, & \text{если } (x_1, \dots, x_n) \in U, \\ \text{не определено,} & \text{если } (x_1, \dots, x_n) \notin U \end{cases}$$

Теорема 3.9. *Множество n -ок U натуральных чисел рекурсивно перечислимо тогда и только тогда, когда его частичная характеристическая функция частично рекурсивна.*

Д о к а з а т е л ь с т в о. Если множество U n -ок натуральных чисел пусто, его частичная характеристическая функция, будучи нигде не определенной, является частично рекурсивной.

Если же множество U n -ок натуральных чисел не пусто, то найдутся такие примитивно рекурсивные функции $\alpha_1(t), \dots, \alpha_n(t)$, что

$$U = \{ (\alpha_1(t), \dots, \alpha_n(t)) \mid t \in N \}.$$

Но тогда

$$\Gamma_{\chi_U^{(p)}} = \{ (\alpha_1(t), \dots, \alpha_n(t), 1) \mid t \in N \}.$$

Поэтому график $\Gamma_{\chi_U^{(p)}}$ частичной характеристической функции $\chi_U^{(p)}$ множества U рекурсивно перечислим, а значит сама частичная характеристическая функцию $\chi_U^{(p)}$ множества U частично рекурсивна.

Обратно, если частичная характеристическая функцию $\chi_U^{(p)}$ множества U частично рекурсивна, то рекурсивно перечислима ее область определения $D(\chi_U^{(p)})$, которая совпадает с множеством U . \square

Доказанная теорема дает еще одну характеристику рекурсивно перечислимых множеств, как множеств, для которых найдется алгоритм, завершающий свою работу только на элементах этого множества (применимый только к элементам этого множества). Исходное определение рекурсивно перечислимого множества давало алгоритм, дающий элементы этого множества в качестве результатов своей работы.

В целом доказанные теоремы могут рассматриваться как свидетельство достаточной естественности понятия рекурсивно перечислимого множества. Еще одним таким свидетельством может служить следующая теорема.

Теорема 3.10. *Для любой частично рекурсивной функции*

$$f(x_1, \dots, x_n, y_1, \dots, y_m)$$

множество

$$U_f = \{ (a_1, \dots, a_n) = (\exists y_1) \dots (\exists y_m) (f(a_1, \dots, a_n, y_1, \dots, y_m) = 0) \}$$

рекурсивно перечислимо.

Д о к а з а т е л ь с т в о. Для любой частично рекурсивной функции $f(x_1, \dots, x_n, y_1, \dots, y_m)$ ее график Γ_f является рекурсивно перечислимым множеством. Если он пуст, то пусто и множество U_f .

В противном случае найдутся такие примитивно рекурсивные функции $\alpha_1(t), \dots, \alpha_n(t), \beta_1(t), \dots, \beta_m(t)$ и $\beta(t)$, что

$$\Gamma_f = \{ (\alpha_1(t), \dots, \alpha_n(t), \beta_1(t), \dots, \beta_m(t), \beta(t)) \}.$$

Но тогда

$$U_f = \{ (a_1, \dots, a_n) \mid (\exists t) (a_1 = \alpha_1(t) \& a_n = \alpha_n(t) \& \beta(t) = 0) \}.$$

Так как функции $\alpha_1(t), \dots, \alpha_n(t)$ и $\beta(t)$ примитивно рекурсивны, то множество U_f рекурсивно перечислимо. \square

При $m = 0$ получаем следствие.

Следствие 1. Для любой частично рекурсивной функции $f(x_1, \dots, x_n)$ множество

$$U_f = \{ (a_1, \dots, a_n) \mid f(a_1, \dots, a_n) = 0 \}$$

рекурсивно перечислимо.

Следствие 2. Любое рекурсивное множество является рекурсивно перечислимым.

Доказательство. Если множество U рекурсивно, то его характеристическая функция $\chi_U(x_1, \dots, x_n)$ рекурсивна. Рекурсивная перечислимость множества U следует из равенства

$$U = \{ (a_1, \dots, a_n) \mid \overline{sg}(\chi_U(a_1, \dots, a_n)) = 0 \}.$$

\square

В качестве следствия теоремы о графике функции получим теорему о нормальной форме С. Клини. Заметим, что позже будет доказан более сильный вариант теоремы С. Клини. А пока докажем следующую теорему.

Теорема 3.11 (Нормальная форма С. Клини). Для любой n -местной частично рекурсивной функции f найдется такая $n+1$ -местная функция F , что

$$f(x_1, \dots, x_n) = l(\mu_t(F(x_1, \dots, x_n, t) = 0)).$$

Доказательство. Если f — n -местная частично рекурсивная функция, то ее график Γ_f рекурсивно перечислим. Поэтому найдется такая примитивно рекурсивная функция $h(x_1, \dots, x_n, y, z)$, что

$$(a_1, \dots, a_n, b) \in \Gamma_f \iff (\exists z)h(a_1, \dots, a_n, b, z) = 0.$$

Значит,

$$f(x_1, \dots, x_n) = l(\mu_t(g(x_1, \dots, x_n, l(t), r(t)) = 0)).$$

Для завершения доказательства теоремы достаточно положить

$$F(x_1, \dots, x_n, t) = g(x_1, \dots, x_n, l(t), r(t)).$$

\square

ГЛАВА III

МАШИНЫ ТЬЮРИНГА

§1. Машины Тьюринга

Существуют различные варианты машин Тьюринга – с одной или с несколькими лентами, с дополнительной входной лентой, с входной, выходной и рабочей лентами, с многомерными лентами и т.д. С точки зрения вычислительных возможностей эти варианты машин Тьюринга эквивалентны. Их особенности начинают играть существенную роль, лишь когда нас интересует не только принципиальная возможность реализации того или иного алгоритма, но и сложные характеристики его выполнения на рассматриваемой модели. Мы рассмотрим в некотором смысле простейший вариант – *машину Тьюринга с одной конечной лентой потенциально бесконечной в обе стороны*. Смысл сказанного станет ясен из дальнейших определений.

Машина Тьюринга T имеет внешний алфавит, или алфавит ленточных символов,

$$\mathcal{A}_T = \{a_0, a_1, \dots, a_n\},$$

внутренний алфавит, или алфавит внутренних состояний,

$$\mathcal{Q}_T = \{q_0, q_1, \dots, q_m\},$$

программу P_T , ленту и рабочую головку. Элементы внешнего алфавита называются просто *символами*, а элементы внутреннего алфавита – *внутренними состояниями*.

Лента машины Тьюринга T разбита на конечное число ячеек, в которые можно записывать символы из внешнего алфавита $\{a_0, a_1, \dots, a_n\}$, поэтому он и называется *алфавитом ленточных символов*. По чисто техническим причинам удобно считать, что среди ленточных символов имеется так называемый *пустой символ*, который традиционно обозначают через a_0 . Ячейки ленты упорядочены слева направо. Мы рассматриваем вариант определения машины Тьюринга, в котором предполагается, что лента *потенциально бесконечна в обе стороны*. Это означает, что в процессе работы машины Тьюринга разрешается добавлять новые пустые ячейки как к правому, так и к левому концам ленты (считается,

что в момент добавления в них записан пустой символ a_0). В принципе можно было бы считать ленту бесконечной, но потребовать, чтобы в каждый момент времени лишь конечное число ее ячеек содержало непустые символы.

Машина Тьюринга выполняет действия в дискретные моменты времени или, как говорят, потактово.

В каждый момент времени **головка** машины Тьюринга T обозревает некоторую ячейку и сама находится в одном из *внутренних состояний* q_0, q_1, \dots, q_m . Состояние q_0 традиционно называется *заключительным состоянием*.

Работа машины Тьюринга состоит из отдельных тактов, выполняемых согласно ее программе P_T .

Программа P_T машины Тьюринга T – это конечный набор команд вида:

$$q_i a_j \longrightarrow q_s a_t D,$$

где $D \in \{L, S, R\}$. При этом каждой паре символов (q_i, a_j) при $i > 0$ соответствует ровно одна команда, в которой перед \longrightarrow стоит слово $q_i a_j$. Эту команду мы будем обозначать через $P_T(i, j)$.

Если в рассматриваемый момент времени головка машины Тьюринга находится в состоянии q_i при $i > 0$, а в обозреваемой ею ячейке находится символ a_j , то машина выполняет команду $P_T(i, j)$ следующим образом:

1) если команда имеет вид

$$q_i a_j \longrightarrow q_s a_t S,$$

то в обозреваемой ячейке символ a_j заменяется на символ a_t , головка переходит в состояние q_s и остается на месте;

2) если команда имеет вид

$$q_i a_j \longrightarrow q_s a_t R \quad \text{или} \quad q_i a_j \longrightarrow q_s a_t L,$$

то в обозреваемой ячейке символ a_j заменяется на символ a_t , головка переходит в состояние q_s и сдвигается в соседнюю справа (соответственно слева) ячейку. При этом, если считывающая головка находилась в самой правой (левой) ячейке, то к ленте добавляется новая пустая ячейка, в которую записан пустой символ a_0 .

Если в результате выполнения некоторой команды *головка машины приходит в заключительное состояние* q_0 , то машина **останавливается**, так как по определению ни одна команда программы не начинается с q_0 .

Полное описание состояния машины Тьюринга в данный момент времени задается словом вида $Aq_i a_j B$, где q_i – состояние головки в данный момент, a_j – символ, записанный в обозреваемой ячейке, а A и B – слова, записанные левее и правее этой ячейки. Такое слово $Aq_i a_j B$ называется **полной конфигурацией** или просто **конфигурацией**, описывающей состояние машины Тьюринга в данный момент. При этом будем говорить, что *машина T находится в конфигурации $Aq_i a_j B$* .

Пусть конфигурация K_T , в которой машина Тьюринга T находится в данный момент времени, имеет вид $K_T = Aq_i a_j B$, где $1 \leq i \leq m$, $0 \leq j \leq n$.

Определим конфигурацию K'_T , в которую машина перейдет в результате применения соответствующей команды $P_T(i, j)$ из программы P_T .

1) Если команда имеет вид $q_i a_j \rightarrow q_s a_t S$, то $K'_T = Aq_s a_t B$.

2) Пусть выполняемая команда имеет вид $q_i a_j \rightarrow q_s a_t R$.

Если слово B непусто, т.е. $B = a_p B'$ при некотором a_p , то

$$K'_T = Aa_t q_s a_p B'.$$

Если же слово B пусто, то $K'_T = Aa_t q_s a_0$.

3) Пусть выполняемая команда $P_T(i, j)$ имеет вид $q_i a_j \rightarrow q_s a_t L$.

Если слово A непусто, т.е. $A = A' a_p$ при некотором a_p , то

$$K'_T = A' q_s a_p a_t B'.$$

Если же слово A пусто, то $K'_T = q_s a_0 a_t B$.

Среди внутренних состояний машины Тьюринга T часто выделяют еще и **начальное** состояние q_1 .

С каждой машиной Тьюринга T связан алгоритм \mathcal{A}_T , преобразующий (перерабатывающий) слова в ее внешнем алфавите \mathcal{A}_T в слова в том же алфавите.

Пусть E – слово во внешнем алфавите \mathcal{A}_T машины Тьюринга T , т.е. $E \in \mathcal{A}_T^*$. Запишем слово E последовательно на ленту машины, начиная со второй ячейки, а в первую ячейку поместим пустой символ a_0 . Поместим головку на первую ячейку и приведем ее в начальное состояние q_1 , т.е. приведем машину Тьюринга T в конфигурацию $q_1 a_0 E$.

Если машина Тьюринга T , начав работать в конфигурации $q_1 a_0 E$, через конечное число тактов работы перейдет в конфигурацию вида $Dq_0 B$, то слово DB называется *результатом применения к слову E алгоритма \mathcal{A}_T* , определяемого машиной Тьюринга T . В противном случае мы говорим, что *алгоритм \mathcal{A}_T к слову E не применим*. Другими словами, мы считаем, что машина T вычисляет некоторую словарную функцию $f_T(x)$, значением которой при $x = E$ является слово DB , причем, эта функция определена только для тех слов E , при которых машина остановится.

Введем некоторые обозначения:

запись $Aq_i a_j B \xRightarrow{T} A'q_p a_l B'$ будет обозначать, что машина Тьюринга T из конфигурации $Aq_i a_j B$ за конечное число шагов (тактов работы) переходит в конфигурацию $A'q_p a_l B'$, и при этом не происходит присоединение пустых ячеек к левому концу ленты (не происходит сдвиг за левый край ленты),

запись $Aq_i a_j B \xRightarrow{T} A'q_p a_l B'$ будет обозначать, что машина Тьюринга T из конфигурации $Aq_i a_j B$ за конечное число шагов (тактов работы) переходит в конфигурацию $A'q_p a_l B'$, и при этом не происходит присоединение пустых ячеек ни к левому, ни к правому концам ленты (не происходит сдвиг ни за левый, ни за правый края ленты).

Машины Тьюринга можно использовать для вычисления функций, аргументы и значения которых пробегают слова в данном алфавите, содержащемся в алфавите ленточных символов. В частности, речь может идти о вычислении *функций от натурального аргумента*.

Определим два понятия вычислимости функции машиной Тьюринга – **вычислимость** и **правильная вычислимость**.

Договоримся о некоторых общепринятых соглашениях: символ a_0 будем обозначать через 0, символ a_1 – через 1, для любой буквы a через a^m будем обозначать слово длины m , все буквы которого – это a , т.е. слово вида $a \dots a$. При этом натуральное число n представляется словом \bar{n} длины n в однобуквенном алфавите $\{1\}$, т.е. словом 1^n .

Пусть f – n -местная (частичная) функция с областью определения $D(f) \subseteq N^n$.

Машина Тьюринга T **вычисляет** функцию f , если выполнены два условия: пусть $\langle b_1, \dots, b_n \rangle$ – произвольный набор натуральных чисел

1) если $\langle b_1, \dots, b_n \rangle \notin D(f)$, то, начав работать в конфигурации

$$q_1 0 1^{b_1} 0 \dots 0 1^{b_n} 0,$$

машина Тьюринга T никогда не придет в заключительное состояние q_0 (машина не остановится, работает бесконечно долго, вечно),

2) если $\langle b_1, \dots, b_n \rangle \in D(f)$, то, начав работать в конфигурации

$$q_1 0 1^{b_1} 0 \dots 0 1^{b_n} 0,$$

машина Тьюринга T через конечное число шагов (тактов работы) придет в заключительное состояние q_0 (остановится), полученная при этом конфигурация имеет вид Cq_0D , и в слово CD символ 1 входит $f(b_1, \dots, b_n)$ раз.

Машина Тьюринга T **правильно вычисляет** функцию f , если выполнены два условия:

пусть $\langle b_1, \dots, b_n \rangle$ – произвольный набор натуральных чисел

1) если $\langle b_1, \dots, b_n \rangle \notin D(f)$, то, начав работать в конфигурации

$$q_1 0 1^{b_1} 0 \dots 0 1^{b_n} 0,$$

машина Тьюринга T никогда не придет в заключительное состояние q_0 (машина не остановится, работает бесконечно долго, вечно),

2) если $\langle b_1, \dots, b_n \rangle \in D(f)$, то, начав работать в конфигурации

$$q_1 0 1^{b_1} 0 \dots 0 1^{b_n} 0,$$

машина Тьюринга T через конечное число шагов (тактов работы) придет в заключительное состояние q_0 (остановится), причем

$$q_1 0 1^{b_1} 0 \dots 0 1^{b_n} 0 \xRightarrow{T} q_0 0 1^{f(b_1, \dots, b_n)} 0 \dots 0.$$

Числовая функция называется **вычислимой по Тьюрингу**, если существует машина Тьюринга, вычисляющая эту функцию.

Позже будет доказано, что следующее утверждение равносильно **Тезису Черча**.

Тезис Тьюринга. Если некоторая функция вычислима в интуитивном смысле, то она вычислима по Тьюрингу.

Тезис Тьюринга связывает интуитивное понятие вычислимой функции с точным математическим понятием вычислимой по Тьюрингу функции, поэтому не может ставиться вопрос о его математическом доказательстве. С этой точки зрения **Тезис Тьюринга** подобен физическому закону и любой естественно-научной гипотезе. Доказать **Тезис Тьюринга** нельзя, однако необходимо его как-то обосновать.

Нашей ближайшей целью будет приведение достаточно убедительных, на наш взгляд, доводов в пользу принятия **Тезиса Тьюринга**, а значит, и равносильного ему **Тезиса Черча**. С этой целью мы разовьем некоторую технику "Тьюрингова программирования", создадим "библиотеку стандартных Тьюринговых программ", разработаем некоторые методы получения новых "Тьюринговых программ" из имеющихся, докажем *правильную вычислимость по Тьюрингу любой частично рекурсивной функции* и частичную рекурсивность каждой вычислимой по Тьюрингу функции. Все это вместе с установленной ранее частичной рекурсивностью важных теоретико-числовых функций может служить весомым аргументом в пользу принятия **Тезиса Тьюринга**, а значит, и равносильного ему **Тезиса Черча**.

Определим две операции над машинами Тьюринга, которые существенно облегчат нам построение новых машин Тьюринга по ранее построенным.

Первая операция – это **композиция** машин Тьюринга.

Пусть T_1 и T_2 – машины Тьюринга с одним и тем же внешним алфавитом \mathcal{A} и внутренними алфавитами $\mathcal{Q}_1 = \{q_0, q_1, \dots, q_m\}$ и $\mathcal{Q}_2 = \{q_0, q_1, \dots, q_p\}$ соответственно.

Тогда **композицией** машин Тьюринга T_1 и T_2 называется машина Тьюринга T с тем же внешним алфавитом \mathcal{A} , внутренним алфавитом $\mathcal{Q} = \{q_0, q_1, \dots, q_m, q_{m+1}, \dots, q_{m+p}\}$ и программой P_T , где

$$P_T = (P_{T_1})_{q_0}[q_{m+1}] \cup (P_{T_2})_{q_1, \dots, q_p}[q_{m+1}, \dots, q_{m+p}].$$

Через $(P_{T_2})_{q_1, \dots, q_p}[q_{m+1}, \dots, q_{m+p}]$ обозначен результат одновременной замены во всех командах программы P_{T_2} q_1 на q_{m+1} , q_2 на q_{m+2} , \dots , q_p на q_{m+p} , а через $(P_{T_1})_{q_0}[q_{m+1}]$ обозначен результат замены во всех командах программы P_{T_1} q_0 на q_{m+1} .

Построенную машину Тьюринга T , являющуюся **композицией** машин T_1 и T_2 , будем обозначать через $T_1 \cdot T_2$, а иногда и просто $T_1 T_2$.

Если машина Тьюринга T начинает работать в конфигурации Zq_iW , где Z и W – слова в ее входном алфавите \mathcal{A} , а $1 \leq i \leq m$, то она "сначала работает как машина Тьюринга T_1 " – если машина Тьюринга T_1 , начав работать в конфигурации Zq_iW , не остановится, то не остановится и машина Тьюринга T , если

же машина Тьюринга T_1 , начав работать в конфигурации Zq_iW , остановится в конфигурации Uq_0V , то дальше машина Тьюринга T "работает как машина Тьюринга T_2 , начав работать в конфигурации Uq_1V ". Более точно, если машина Тьюринга T_2 , начав работать в конфигурации Uq_1V , не остановится, то не остановится и машина Тьюринга T , если же машина Тьюринга T_2 , начав работать в конфигурации Uq_1V , остановится в конфигурации Xq_0Y , то машина Тьюринга T остановится в этой же конфигурации. Если же $m + 1 \leq i \leq m + p$, то машина Тьюринга T "работает как машина Тьюринга T_2 ".

Приступаем к построению машин Тьюринга, вычисляющих необходимые для дальнейшего функции. У всех рассматриваемых ниже машин Тьюринга T будет один и тот же внешний алфавит $\mathcal{A}_T = \{0, 1\}$, а их внутренний алфавит будет виден из системы команд.

Перенос нуля. Построим машину Тьюринга T_0 , выполняющую преобразование

$$q_1 001^a 0 \mid \xRightarrow{T} q_0 01^a 00.$$

$$\begin{array}{ll} q_1 0 \longrightarrow q_2 0R, & q_2 0 \longrightarrow q_3 1S, \\ q_3 1 \longrightarrow q_3 1R, & q_3 0 \longrightarrow q_4 0L, \\ q_4 1 \longrightarrow q_5 0L, & q_5 1 \longrightarrow q_5 1L, \\ q_5 0 \longrightarrow q_0 0S. \end{array}$$

Правый сдвиг. Построим машину Тьюринга Rs (*right – shift*), выполняющую преобразование

$$q_1 01^a 0 \mid \xRightarrow{Rs} 01^a q_0 0.$$

$$\begin{array}{ll} q_1 0 \longrightarrow q_2 0R, & q_2 1 \longrightarrow q_2 1R, \\ q_2 0 \longrightarrow q_0 0S. \end{array}$$

Левый сдвиг. Построим машину Тьюринга Ls (*left – shift*), выполняющую преобразование

$$01^a q_1 0 \mid \xRightarrow{Ls} q_0 01^a 0.$$

$$\begin{array}{ll} q_1 0 \longrightarrow q_2 0L, & q_2 1 \longrightarrow q_2 1L, \\ q_2 0 \longrightarrow q_0 0S. \end{array}$$

Транспозиция. Построим машину Тьюринга Tr (*transposition*), выполняющую преобразование

$$01^a q_1 01^b 0 \mid \xRightarrow{Tr} 01^b q_0 01^a 0.$$

Конечно, программу машины Тьюринга Tr можно было бы выписать непосредственно команда за командой, и это хорошее упражнение для читателя, однако мы на этом несложном примере продемонстрируем некоторые важные

для дальнейшего приема построения искомых машин Тьюринга по имеющимся в соответствии с заранее составленным планом. При реализации этого плана будет полезна еще одна операция над машинами Тьюринга – **разветвление**.

Разветвление машин Тьюринга T_0 , T_1 и T_2 по паре состояний $\langle q_s, q_t \rangle$.

Пусть T_0 , T_1 и T_2 – машины Тьюринга с одним и тем же внешним алфавитом \mathcal{A} и внутренними алфавитами $\mathcal{Q}_0 = \{q_0, q_1, \dots, q_k\}$, $\mathcal{Q}_1 = \{q_0, q_1, \dots, q_m\}$ и $\mathcal{Q}_2 = \{q_0, q_1, \dots, q_p\}$ соответственно, а q_s и q_t – два выделенных состояния машины Тьюринга T_0 .

Требуется построить машину Тьюринга T с тем же внешним алфавитом \mathcal{A} , с внутренним алфавитом, содержащим внутренний алфавит машины T_0 и работающую следующим образом:

начав работать в конфигурации Uq_iW , где U , W – слова в ее входном алфавите \mathcal{A} , а q_i – внутреннее состояние машины Тьюринга T_0 , она ”сначала работает как машина Тьюринга T_0 ”, при этом, если в ходе вычислений она приходит в состояние q_s и находится в этот момент в конфигурации Xq_sY , то дальше она работает так, ”как работала бы машина Тьюринга T_1 , начав работать в конфигурации Xq_1Y ”, если же в ходе вычислений она приходит в состояние q_t и находится в этот момент в конфигурации Xq_tY , то дальше она работает так, ”как работала бы машина Тьюринга T_2 , начав работать в конфигурации Xq_1Y ”.

В качестве машины T можно взять машину Тьюринга с тем же внешним алфавитом \mathcal{A} , внутренним алфавитом

$$\mathcal{Q} = \{q_0, q_1, \dots, q_k, q_{k+1}, \dots, q_{k+m-1}, q_{k+m}, \dots, q_{k+m+p-2}\}$$

и программой P_T , где

$$P_T = (P_{T_0})_{(\hat{s}, \hat{t})} \cup (P_{T_1})_{q_1, q_2, \dots, q_m} [q_s, q_{k+1}, \dots, q_{k+m-1}] \cup (P_{T_2})_{q_1, q_2, \dots, q_p} [q_t, q_{k+m}, \dots, q_{k+m+p-2}].$$

Через $(P_{T_0})_{(\hat{s}, \hat{t})}$ обозначен результат удаления из программы машины Тьюринга P_{T_0} всех команд $P_{T_0}(s, j)$ и $P_{T_0}(t, j)$, т.е. команд с левой частью вида $q_s a_j$ или $q_t a_j$. Смысл остальных обозначений такой же, как и при построении композиции машин Тьюринга.

Разветвление машин Тьюринга T_0 , T_1 и T_2 по паре состояний $\langle q_s, q_t \rangle$ будем обозначать через

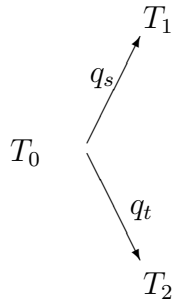


Рис. Т1.

План построения машины Тьюринга Tr (*transposition*), выполняющей преобразование

$$01^a q_1 01^b 0 \mid \xRightarrow{Tr} 01^b q_0 01^a 0.$$

1) Строим машину T_1 , выполняющую преобразование

$$q_1 01^a 0 \mid \xRightarrow{T_1} q_0 1^a 00$$

$$\begin{array}{ll} q_1 0 \longrightarrow q_2 0R, & q_1 1 \longrightarrow q_2 1S, \\ q_2 1 \longrightarrow q_2 1R, & q_2 0 \longrightarrow q_3 0L, \\ q_3 1 \longrightarrow q_4 0L, & q_3 0 \longrightarrow q_0 0S, \\ q_4 1 \longrightarrow q_4 1L, & q_4 0 \longrightarrow q_0 1S. \end{array}$$

2) Строим машину T , которая выполняет преобразование

$$0q_1 1^b 01^a 0 \mid \xRightarrow{T} 01^b q_0 01^a 0,$$

а при $a - t > 0$ – преобразование

$$01^{a-t} q_1 1^b 01^t 0 \mid \xRightarrow{T} 01^{a-t-1} q_1 1^b 01^{t+1} 0.$$

$$\begin{array}{ll} q_1 0 \longrightarrow q_2 0L, & q_1 1 \longrightarrow q_2 1L, \\ q_2 0 \longrightarrow q_3 0R, & q_3 1 \longrightarrow q_3 1R, \\ q_3 0 \longrightarrow q_0 0S, & q_2 1 \longrightarrow q_4 0R, \\ q_4 1 \longrightarrow q_4 1R, & q_4 0 \longrightarrow q_5 1L, \\ q_5 0 \longrightarrow q_1 0S, & q_5 1 \longrightarrow q_6 0L, \\ q_6 1 \longrightarrow q_6 1L, & q_6 0 \longrightarrow q_1 1S. \end{array}$$

В качестве машины Тьюринга Tr , выполняющей транспозицию, можно взять композицию $T_1 \cdot T$ машин Тьюринга T_1 и T .

Начав работать в конфигурации $01^a q_1 01^b 0$, машина Тьюринга Tr , работая как машина Тьюринга T_1 , выполнит переход вида

$$01^a q_1 01^b 0 \mid \xRightarrow{Tr} 01^a q_t 1^b 00.$$

Так как $01^a q_1 1^b 00 = 01^a q_1 1^b 01^0 0$, то машина Тьюринга T_1 "будет переносить 1, стоящие слева от головки, в выделенное место между нулями" пока не возникнет конфигурация $0q_1 1^b 01^a 0$, которая будет переведена в заключительную конфигурацию $01^b q_0 01^a 0$.

Удвоение. Построим машину Тьюринга D , выполняющую преобразование

$$q_1 01^a 0 \xRightarrow{D} q_0 01^a 01^a 0.$$

Построим машину Тьюринга D , выполняющую при $a > 0$ преобразование

$$q_1 01^a 01^b 01^c 0 \xRightarrow{T} q_1 01^{a-1} 01^{b+1} 01^{c+1} 0,$$

а при $a = 0$ – преобразование

$$q_1 001^b 01^c 0 \xRightarrow{T} q_0 01^b 01^c 0.$$

$$\begin{array}{ll} q_1 0 \longrightarrow q_2 0R, & q_2 0 \longrightarrow q_3 1S, \\ q_3 1 \longrightarrow q_3 1R, & q_3 0 \longrightarrow q_4 0R, \\ q_4 1 \longrightarrow q_4 1R, & q_4 0 \longrightarrow q_5 0L, \\ q_5 1 \longrightarrow q_6 0L, & q_5 0 \longrightarrow q_7 0L, \\ q_6 1 \longrightarrow q_6 1L, & q_6 0 \longrightarrow q_7 1L, \\ q_7 1 \longrightarrow q_8 0L, & q_8 1 \longrightarrow q_8 1L, \\ q_8 0 \longrightarrow q_0 0S, & q_2 1 \longrightarrow q_9 1S, \\ q_9 1 \longrightarrow q_9 1R, & q_9 0 \longrightarrow q_{10} 0L, \\ q_{10} 1 \longrightarrow q_{11} 0R, & q_{11} 0 \longrightarrow q_{12} 1S, \\ q_{12} 1 \longrightarrow q_{12} 1R, & q_{12} 0 \longrightarrow q_{13} 0R, \\ q_{13} 1 \longrightarrow q_{13} 1R, & q_{13} 0 \longrightarrow q_{14} 1S, \\ q_{14} 1 \longrightarrow q_{14} 1L, & q_{14} 0 \longrightarrow q_{15} 0L, \\ q_{15} 1 \longrightarrow q_{15} 1L, & q_{15} 0 \longrightarrow q_{16} 0L, \\ q_{16} 1 \longrightarrow q_{16} 1L, & q_{16} 0 \longrightarrow q_1 0S. \end{array}$$

Если применить машину T к начальной конфигурации $q_1 01^a 0 = q_1 01^a 01^0 01^0 0$, то через конечное число шагов получим заключительную конфигурацию $q_0 01^a 01^a 0$.

Циклический сдвиг. Для произвольного натурального числа n построим машину Тьюринга S_n , выполняющую преобразование

$$q_1 01^{b_1} 01^{b_2} 0 \dots 01^{b_n} \mid \xRightarrow{S_n} q_0 01^{b_2} 0 \dots 01^{b_n} 01^{b_1}.$$

В качестве машины Тьюринга S_n можно взять

$$(Rs \cdot Tr)^{n-1} \cdot (Ls)^{n-1}.$$

Копирование. Для произвольного натурального числа n построим машину Тьюринга C_n , выполняющую преобразование

$$q_1 01^{b_1} 01^{b_2} 0 \dots 01^{b_n} \xRightarrow{C_n} q_0 01^{b_1} 01^{b_2} 0 \dots 01^{b_n} 01^{b_1} 01^{b_2} 0 \dots 01^{b_n}.$$

Построение машины Тьюринга C_n выполним индукцией по n .

Полагаем $C_1 = D$. Если машина Тьюринга C_n уже построена, полагаем

$$C_{n+1} = (Rs)^n \cdot D \cdot (Ls)^n \cdot (S_{n+2})^n \cdot (Rs)^2 \cdot C_n \cdot (Ls)^2 \cdot S_{2n+2} \cdot S_{n+1}.$$

Лемма 1.1. *Исходные (простейшие) функции правильно вычислимы по Тьюрингу.*

Д о к а з а т е л ь с т в о. Машина Тьюринга с программой

$$\begin{array}{ll} q_1 0 \longrightarrow q_2 0 R, & q_2 1 \longrightarrow q_2 1 R, \\ q_2 0 \longrightarrow q_3 1 S, & q_3 1 \longrightarrow q_3 1 L, \\ q_3 0 \longrightarrow q_0 0 S & \end{array}$$

правильно вычисляет функцию следования $s(x)$, а машина Тьюринга O_1 с программой

$$\begin{array}{ll} q_1 0 \longrightarrow q_2 0 R, & q_2 1 \longrightarrow q_2 1 R, \\ q_2 0 \longrightarrow q_3 0 L, & q_3 1 \longrightarrow q_3 0 L, \\ q_3 0 \longrightarrow q_0 0 S & \end{array}$$

правильно вычисляет нулевую функцию $0(x)$.

Функцию проектирования $U_m^n(x_1, \dots, x_n) = x_m$ правильно вычисляет машина Тьюринга

$$(S_m)^{m-1} \cdot (Rs)^{n-1} \cdot (O_1 \cdot Ls)^{n-1}.$$

□

Лемма 1.2. *Если n -местная функция f получена применением оператора суперпозиции к правильно вычислимым m -местной функции g и m n -местным функциям f_1, \dots, f_m , то и функция f правильно вычислима.*

Д о к а з а т е л ь с т в о. Пусть машина Тьюринга G правильно вычисляет m -местную функцию g , а машины F_1, \dots, F_m правильно вычисляют n -местные функции f_1, \dots, f_m . Тогда машина

$$\begin{array}{l} C_n \cdot (Rs)^n \cdot F_1 \cdot (Ls)^n \cdot (S_{n+1})^n \cdot Rs \cdot C_n \cdot (Rs)^n \cdot F_2 \cdot (Ls)^n \cdot (S_{n+1})^n \cdot Rs \cdot \dots \\ C_n \cdot (Rs)^n \cdot F_m \cdot (Ls)^n \cdot (S_{n+1})^n \cdot Rs \cdot (Ls)^{m-1} \cdot G \end{array}$$

правильно вычисляет функцию $f = S(g; f_1, \dots, f_m)$.

□

Лемма 1.3. *Если $n+1$ -местная функция f получена применением оператора примитивной рекурсии к правильно вычислимым n -местной функции g и $n+2$ -местной функции h , то и функция f правильно вычислима.*

Д о к а з а т е л ь с т в о. Пусть машина Тьюринга G правильно вычисляет n -местную функцию g , а машина H правильно вычисляет функцию h . Построим машину Тьюринга T , которая будет правильно вычислять функцию $f = PR(g; h)$.

Обозначим через T_1 следующую машину Тьюринга

$$(S_{n+1})^n \cdot (Rs) \cdot (C_{n+1}) \cdot (Rs)^{n+1} \cdot G \cdot (Ls)^{n+2} \cdot (S_{n+2}) \cdot (Rs)^{n+2}.$$

Непосредственная проверка с выписыванием "узловых" промежуточных конфигураций показывает, что

$$q_1 01^{b_1} 0 \dots 01^{b_n} 01^b 0 \xRightarrow{T_1} 01^{b_1} 0 \dots 01^{b_n} 01^0 01^{b-0} q_0 01^{f(b_1, \dots, b_n, 0)}$$

Конечно,

$$01^{b_1} 0 \dots 01^{b_n} 01^0 01^{b-0} q_0 01^{f(b_1, \dots, b_n, 0)} = 01^{b_1} 0 \dots 01^{b_n} 001^b q_0 01^{g(b_1, \dots, b_n)},$$

но мы выбрали указанное представление, чтобы лучше был ясен ход вычислений.

Построим машину Тьюринга T_2 , выполняющую переход

$$01^{b_1} 0 \dots 01^{b_n} 01^b 01^{b-b} q_1 01^{f(b_1, \dots, b_n, b)} \xRightarrow{T_2} q_0 01^{f(b_1, \dots, b_n, b)} 0.$$

Машину Тьюринга T_2 строим как композицию ранее построенных машин и попутно выписываем "ключевые" конфигурации

$$\begin{aligned} & 01^{b_1} 0 \dots 01^{b_n} 01^b 01^{b-b} q_1 01^{f(b_1, \dots, b_n, b)} \\ (Ls)^{n+2} : & q_0 1^{b_1} 0 \dots 01^{b_n} 01^b 01^{b-b} 01^{f(b_1, \dots, b_n, b)} \\ (S_{n+3})^{n+2} : & q_0 1^{f(b_1, \dots, b_n, b)} 01^{b_1} 0 \dots 01^{b_n} 01^b 0 \\ (Rs)^{n+1} : & 01^{f(b_1, \dots, b_n, b)} 01^{b_1} 0 \dots 01^{b_n} q_0 1^b 0 \\ (O_1 \cdot Ls)^n : & q_0 01^{f(b_1, \dots, b_n, b)} 0. \end{aligned}$$

Построим машину Тьюринга T_3 , выполняющую при $b - i > 0$ переход

$$\begin{aligned} 01^{b_1} 0 \dots 01^{b_n} 01^i 01^{b-i} q_1 01^{f(b_1, \dots, b_n, i)} & \xRightarrow{T_3} \\ & 01^{b_1} 0 \dots 01^{b_n} 01^{i+1} 01^{b-(i+1)} q_0 01^{f(b_1, \dots, b_n, i+1)} 0. \end{aligned}$$

Машину Тьюринга T_3 строим как композицию ранее построенных машин и попутно выписываем "ключевые" конфигурации

$$\begin{aligned} & 01^{b_1} 0 \dots 01^{b_n} 01^i 01^{b-i} q_1 01^{f(b_1, \dots, b_n, i)} \\ (Ls)^{n+2} : & q_0 1^{b_1} 0 \dots 01^{b_n} 01^i 01^{b-i} 01^{f(b_1, \dots, b_n, i)} 0 \\ (S_{n+3})^{n+1} : & q_0 1^{b-i} 01^{f(b_1, \dots, b_n, i)} 01^{b_1} 0 \dots 01^{b_n} 01^i 0 \\ (Rs)^2 \cdot C_{n+1} \cdot Ls : & 01^{b-i} q_0 1^{f(b_1, \dots, b_n, i)} 01^{b_1} 0 \dots 01^{b_n} 01^i 01^{b_1} 0 \dots 01^{b_n} 01^i 0 \\ S_{2n+3} \cdot (Rs)^{n+1} \cdot H : & 01^{b-i} 01^{b_1} 0 \dots 01^{b_n} 01^i 0 q_0 1^{f(b_1, \dots, b_n, i+1)} \\ (Ls)^{n+2} \cdot S_{n+2} \cdot (Rs)^{n+1} : & 01^{b_1} 0 \dots 01^{b_n} 01^i q_0 1^{b-i} 01^{f(b_1, \dots, b_n, i+1)} \\ T' : & 01^{b_1} 0 \dots 01^{b_n} 01^{i+1} 01^{b-(i+1)} q_0 01^{f(b_1, \dots, b_n, i+1)}, \end{aligned}$$

где через T' обозначена машина, выполняющая при $b - i > 0$ переход

$$01^i q_1 01^{b-i} 0 \mid \xRightarrow{T'} 01^{i+1} 01^{b-(i+1)} q_0 0$$

$$\begin{array}{ll} q_1 0 \longrightarrow q_2 1R, & q_2 1 \longrightarrow q_3 0R, \\ q_3 1 \longrightarrow q_3 1R, & q_3 0 \longrightarrow q_3 0S. \end{array}$$

Для построения разветвления машин T_2 и T_3 построим "управляющую машину" T_4 , которая из конфигурации

$$01^{b_1}0 \dots 01^{b_n}01^i 01^{b-i} q_1 01^{f(b_1, \dots, b_n, i)}$$

переходит в состояние q_2 при $b - i = 0$ и в состояние q_3 при $b - i > 0$:

$$\begin{array}{ll} q_1 0 \longrightarrow q_4 0L, & q_4 1 \longrightarrow q_3 1R, \\ q_4 0 \longrightarrow q_2 0R. \end{array}$$

Построим машину Тьюринга T_5 , являющуюся *разветвлением машин Тьюринга* T_4 , T_2 и T_3 по паре состояний $\langle q_2, q_3 \rangle$ с *защелкиванием* T_3 на T_4 . Это означает, что сначала машина T_5 работает как T_4 . Если T_4 приходит в состояние q_2 , то дальше T_5 работает как T_2 и после прихода T_2 в состояние q_0 машина T_5 останавливается. Если T_4 приходит в состояние q_3 , то дальше T_5 работает как T_3 и после прихода T_3 в состояние q_0 вновь начинает работать машина T_4 .

Для получения машины T_5 сначала строим машину T'_5 , являющуюся *разветвлением машин Тьюринга* T_4 , T_2 и T_3 по паре состояний $\langle q_2, q_3 \rangle$, которую мы обозначаем через

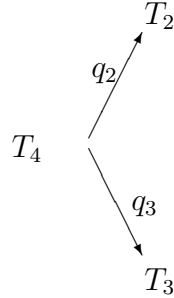


Рис. Т2.

В программе машины T'_5 вхождения состояния q_0 , появившиеся из вхождений этого состояния q_0 в программу машины T_3 при построении разветвления T'_5 машин Тьюринга T_4 , T_2 и T_3 по паре состояний $\langle q_2, q_3 \rangle$, заменим на начальное состояние q_1 машины T_4 .

Для лучшего понимания сказанного можно обратиться к рисунку Т3.

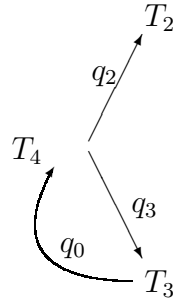


Рис. Т3.

В качестве машины Тьюринга T , правильно вычисляющей функцию $f = PR(g; h)$ можно взять машину $T_1 \cdot T_5$.

Для лучшего понимания сказанного можно обратиться к рисунку Т4.

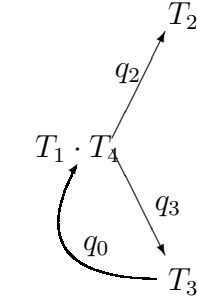


Рис. Т4.

□

Сделаем одно замечание по поводу μ -оператора. Равенство

$$f(x_1, \dots, x_n) = \mu_y(g(x_1, \dots, x_n, y) = h(x_1, \dots, x_n, y))$$

можно заменить равенством

$$f(x_1, \dots, x_n) = \mu_y(|g(x_1, \dots, x_n, y) - h(x_1, \dots, x_n, y)| = 0),$$

поэтому всегда можно считать, что задание функции μ -оператором имеет вид

$$f(x_1, \dots, x_n) = \mu_y(g(x_1, \dots, x_n, y) = 0).$$

Лемма 1.4. Если n -местная функция f получена применением μ -оператора к правильно вычислимой $n + 1$ -местной функции g , то и функция f правильно вычислима.

Д о к а з а т е л ь с т в о. Пусть машина Тьюринга G правильно вычисляет $n + 1$ -местную функцию g . Построим машину Тьюринга T , которая будет правильно вычислять функцию $f = \mu_y g$.

Обозначим через T_1 следующую машину Тьюринга $C_{n+1} \cdot (Rs)^{n+1}$. Тогда

$$q_1 01^{b_1} 0 \dots 01^{b_n} 0 \xRightarrow{T_1} 01^{b_1} 0 \dots 01^{b_n} 01^0 q_0 01^{b_1} 0 \dots 01^{b_n} 01^0.$$

Тогда машина Тьюринга G выполняет переход

$$01^{b_1} 0 \dots 01^{b_n} 01^i q_1 01^{b_1} 0 \dots 01^{b_n} 01^i \xRightarrow{G} 01^{b_1} 0 \dots 01^{b_n} 01^i q_0 01^{g(b_1, \dots, b_n, i)} 0.$$

Строим машину Тьюринга T_2 , выполняющую при $c = 0$ переход

$$01^{b_1} 0 \dots 01^{b_n} 01^i q_1 01^c 0 \xRightarrow{T_2} q_0 01^i 0.$$

В качестве T_2 можно взять следующую машину Тьюринга

$$(Ls)^{n+1} \cdot (S_{n+1})^n \cdot (Ls)^{n+1} \cdot (O_1 \cdot Ls)^{n+1}.$$

Строим машину Тьюринга T_3 , выполняющую при $c > 0$ переход

$$01^{b_1}0 \dots 01^{b_n}01^i q_1 01^c 0 \xRightarrow{T_2} 01^{b_1}0 \dots 01^{b_n}01^{i+1} q_0 01^{b_1}0 \dots 01^{b_n}01^{i+1}0.$$

В качестве T_3 можно взять следующую машину Тьюринга

$$O_1 \cdot Ls \cdot T_s \cdot (Ls)^n \cdot C_{n+1} \cdot (Rs)^{n+1},$$

где T_s – машина Тьюринга, правильно вычисляющая функцию следования $s(x)$.

Для построения разветвления машин T_2 и T_3 по паре состояний $\langle q_2, q_3 \rangle$ построим "управляющую машину" T_4 , которая из конфигурации

$$01^{b_1}0 \dots 01^{b_n}01^i q_1 01^c 0$$

переходит в состояние q_2 при $c = 0$ и в состояние q_3 при $c > 0$:

$$\begin{aligned} q_1 0 &\longrightarrow q_4 0 R, & q_4 1 &\longrightarrow q_2 1 L, \\ q_4 0 &\longrightarrow q_3 0 L. \end{aligned}$$

Построим машину Тьюринга T_5 , являющуюся композицией G и разветвления машин Тьюринга T_4 , T_2 и T_3 по паре состояний $\langle q_2, q_3 \rangle$ с заикливанием T_3 на G . Это означает, что сначала машина T_5 работает как G , а затем как T_4 . Если T_4 приходит в состояние q_2 , то дальше T_5 работает как T_2 и после прихода T_2 в состояние q_0 машина T_5 останавливается. Если T_4 приходит в состояние q_3 , то дальше T_5 работает как T_3 и после прихода T_3 в состояние q_0 вновь начинает работать машина G .

Для получения машины T_5 сначала строим машину T'_5 , являющуюся разветвлением машин Тьюринга T_4 , T_2 и T_3 по паре состояний $\langle q_2, q_3 \rangle$, которую мы обозначаем через

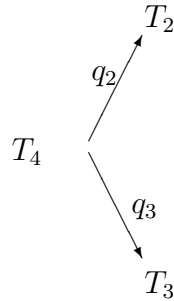


Рис. Т5.

Затем строим композицию $G \cdot T'_5$ машин G и T'_5 , после этого в программе машины $G \cdot T'_5$ вхождения состояния q_0 , появившиеся из вхождений этого состояния

q_0 в программу машины T_3 при построении разветвления T'_5 машин Тьюринга T_4 , T_2 и T_3 по паре состояний $\langle q_2, q_3 \rangle$, а затем композиции $G \cdot T'_5$, заменим на начальное состояние q_1 машины G .

Для лучшего понимания сказанного можно обратиться к рисунку Т6.

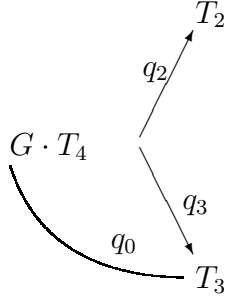


Рис. Т6.

В качестве машины Тьюринга T , правильно вычисляющей функцию $f = \mu_y$ можно взять машину $T_1 \cdot T_5$.

Для лучшего понимания сказанного можно обратиться к рисунку Т7

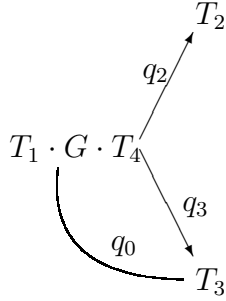


Рис. Т7.

□

Из доказанных лемм сразу следует следующая фундаментальная теорема.

Теорема 1.1. *Любая частично рекурсивная функция правильно вычислима по Тьюрингу.*

Справедлива и обратная теорема.

Теорема 1.2. *Любая вычислимая по Тьюрингу функция частично рекурсивна.*

Заметим, что в теореме речь идет просто о вычислимости, а не обязательно о правильной вычислимости. Непосредственным следствием этих двух теорем будет следующая теорема.

Теорема 1.3. *Любая вычислимая по Тьюрингу функция правильно вычислима.*

Для доказательства двух последних теорем нам потребуется выполнить некоторую работу по *арифметизации теории машин Тьюринга*.

§2. Арифметизация теории машин Тьюринга

Цель этого параграфа – *арифметизировать теорию машин Тьюринга*, заменив каждую из них соответствующим номером. В качестве следствия будут получены *теоремы об алгоритмической неразрешимости проблем применимости и самоприменимости для машин Тьюринга*, а это, в свою очередь, станет отправной точкой на пути установления *алгоритмической неразрешимости ряда проблем в алгебре, математической логике, теории чисел* и т.д.

Будем считать, что внешний алфавит $\mathcal{A}_T = \{a_0, \dots, a_n\}$ любой машины Тьюринга T является подмножеством множества

$$\mathcal{A} = \{a_0, a_1, \dots, a_n, \dots\},$$

а ее внутренний алфавит $\mathcal{Q}_T = \{q_0, q_1, \dots, q_m\}$ – подмножеством множества

$$\mathcal{Q} = \{q_0, q_1, \dots, q_m, \dots\}.$$

Так как состояние любой машины Тьюринга T в каждый момент времени описывается конфигурацией вида $Aq_i a_j B$, где A и B – слова в алфавите \mathcal{Q} , то с каждым словом W в этом алфавите связываются два номера $n_l(W)$ и $n_r(W)$:

$$n_l(a_{i_0} a_{i_1} \dots a_{i_{t-1}} a_{i_t}) = p_0^{i_t} p_1^{i_{t-1}} \dots p_{t-1}^{i_1} p_t^{i_0},$$

$$n_r(a_{i_0} a_{i_1} \dots a_{i_{t-1}} a_{i_t}) = p_0^{i_0} p_1^{i_1} \dots p_{t-1}^{i_{t-1}} p_t^{i_t}.$$

В качестве номера конфигурации $K = Aq_i a_j B$ возьмем число

$$\alpha(K) = 2^{n_l(A)} \cdot 3^i \cdot 5^j \cdot 7^{n_r(B)},$$

в качестве номера команды $P_T(i, j)$ вида

$$q_i a_j \Longrightarrow q_s a_t D$$

возьмем число

$$\beta(P_T(i, j)) = p_{c(i, j)}^{2^s \cdot 3^t \cdot 5^{n(D)}},$$

где $n(S) = 0$, $n(L) = 1$ и $n(R) = 2$.

Номером $n(T)$ машины Тьюринга T назовем произведение номеров $\beta(P_T(i, j))$ всех ее команд.

Покажем, что предикат $Tm(x)$ – “ x – номер некоторой машины Тьюринга” примитивно рекурсивен.

Рассмотрим две функции $I(n)$ и $J(n)$, дающие по номеру n машины Тьюринга T число элементов в ее внутреннем и внешнем алфавитах.

$$I(n) = \max_{i \leq n} (\exists j)_{j \leq n} ((p(c(i, j)) \mid n) \vee \exp(0, \exp(j, n)) = i),$$

$$J(n) = \max_{j \leq n} (\exists i)_{i \leq n} ((p(c(i, j)) \mid n) \vee \exp(1, \exp(i, n)) = j).$$

Для установления *примитивной рекурсивности* функций $I(n)$ и $J(n)$ достаточно заметить, что для любого предиката $P(x_1, \dots, x_n, y, t)$ выполняется равенство

$$\max_{t \leq y} P(x_1, \dots, x_n, y, t) = y - \min_{t \leq y} P(x_1, \dots, x_n, y, y - t).$$

Примитивная рекурсивность предиката $Tm(x)$ — " x — номер некоторой машины Тьюринга" следует из эквивалентности

$$\begin{aligned} Tm(x) \iff & (\forall j)_{j \leq x} \neg(p(c(0, j)) | x) \& \\ & (\forall i)_{1 \leq i \leq I(x)} (\forall j)_{i \leq J(x)} (p(c(i, j)) | x) \& \\ & (\forall y)_{y \leq x} (\exp(y, x) = 0 \vee (\exists i)_{1 \leq i \leq I(x)} (\exists j)_{i \leq J(x)} y = c(i, j)) \& \\ & (\forall i)_{1 \leq i \leq I(x)} (\forall j)_{i \leq J(x)} (\exp(2, \exp(c(i, j), x)) \leq 2 \& \\ & (\forall y)_{y \leq x} \exp(y + 3, \exp(c(i, j), x)) = 0). \end{aligned}$$

Нашей ближайшей целью будет *построение примитивно рекурсивных функций, описывающих процесс вычислений, выполняемых машиной Тьюринга.*

Начнем мы с построения для произвольного натурального числа n n -местной функции $\delta_n(x_1, \dots, x_n)$, которая для произвольного набора (a_1, \dots, a_n) натуральных чисел дает *номер начальной конфигурации*

$$q_1 0 1^{a_1} 0 \dots 0 1^{a_n} 0,$$

т.е. должно быть выполнено равенство

$$\delta_n(a_1, \dots, a_n) = \alpha(q_1 0 1^{a_1} 0 \dots 0 1^{a_n} 0).$$

Обозначим через B слово $1^{a_1} 0 \dots 0 1^{a_n} 0$. Тогда

$$\begin{aligned} n_r(B) = & \prod_{i=0}^{a_1-1} p_i \cdot \prod_{i=a_1+1}^{a_1+a_2} p_i \cdot \prod_{i=a_1+a_2+2}^{a_1+a_2+a_3+1} p_i \cdot \\ & \dots \\ & \prod_{i=a_1+a_2+a_3+\dots+a_{n-1}+a_n+n-2}^{a_1+a_2+a_3+\dots+a_{n-1}+a_n+n-1} p_i. \end{aligned}$$

При этом считаем, что $\prod_{i=0}^{0-1} p_i = 1$. Рассмотрим функцию

$$\begin{aligned} \gamma_r(x_1, \dots, x_n) = & \prod_{i=0}^{x_1-1} p_i \cdot \prod_{i=x_1+1}^{x_1+x_2} p_i \cdot \prod_{i=x_1+x_2+2}^{x_1+x_2+x_3+1} p_i \cdot \\ & \dots \\ & \prod_{i=x_1+x_2+x_3+\dots+x_{n-1}+x_n+n-1}^{x_1+x_2+x_3+\dots+x_{n-1}+x_n+n-2} p_i. \end{aligned}$$

Ясно, что $\gamma_r(x_1, \dots, x_n)$ – примитивно рекурсивная функция.

Так как

$$\alpha(q_1 0 1^{a_1} 0 \dots 0 1^{a_n} 0) = 2^0 3^1 5^0 7^{n_r(B)},$$

то

$$\delta_n(x_1, \dots, x_n) = 3 \cdot 7^{\gamma_r(x_1, \dots, x_n)}.$$

Построим примитивно рекурсивную функцию $S_1(i, j, u, v, s, t, \varepsilon)$ такую, что если команда $P_T(i, j)$ машины Тьюринга T имеет вид

$$q_i a_j \implies q_s a_t D,$$

$a(Aq_i a_j B)_T'$ – конфигурация, полученная из конфигурации $Aq_i a_j B$ в результате выполнения этой команды, то

$$S_1(i, j, n_l(A), n_r(B), s, t, n(D)) = \alpha((Aq_i a_j B)_T').$$

Если $n(D) = 0$, то $D = S$. Поэтому $(Aq_i a_j B)_T' = Aq_s a_t B$ и мы полагаем

$$S_1(i, j, u, v, s, t, 0) = 2^u \cdot 3^s \cdot 5^t \cdot 7^v.$$

Если $n(D) = 1$, то $D = L$. Поэтому $(Aq_i a_j B)_T' = A'q_s a_\alpha a_t B$ при $A = A'a_\alpha$ и $(Aq_i a_j B)_T' = q_s a_0 a_t B$ при пустом A .

Пологаем

$$S_1(i, j, u, v, s, t, 1) = 2^{\prod_{i=0}^u p_i^{\exp(i+1, u)}} \cdot 3^s \cdot 5^{\exp(0, u)} \cdot 7^{p_0^t \cdot \prod_{i=0}^v p_{i+1}^{\exp(i, v)}}.$$

Если $n(D) = 2$, то $D = R$. Поэтому $(Aq_i a_j B)_T' = Aa_t q_s a_\alpha B'$ при $B = a_\alpha B'$ и $(Aq_i a_j B)_T' = Aa_t q_s a_0$ при пустом B .

Пологаем

$$S_1(i, j, u, v, s, t, 2) = 2^{p_0^t \cdot \prod_{i=0}^u p_{i+1}^{\exp(i, u)}} \cdot 3^s \cdot 5^{\exp(0, v)} \cdot 7^{\prod_{i=0}^v p_i^{\exp(i+1, v)}}.$$

Пологаем

$$S_1(i, j, u, v, s, t, \varepsilon) = S_1(i, j, u, v, s, t, 0) \cdot \overline{sg}(\varepsilon) + \\ S_1(i, j, u, v, s, t, 1) \cdot \overline{sg}(|\varepsilon - 1|) + S_1(i, j, u, v, s, t, 2) \cdot \overline{sg}(|\varepsilon - 2|).$$

Построим примитивно рекурсивную функцию $S_2(n, i, j, u, v)$ такую, что если $n = n(T)$ – номер машины Тьюринга T , в программу которой входит команда $P_T(i, j)$, имеющая вид

$$q_i a_j \implies q_s a_t D,$$

$a(Aq_i a_j B)_T'$ – конфигурация, полученная из конфигурации $Aq_i a_j B$ в результате выполнения этой команды, то

$$S_2(n, i, j, n_l(A), n_r(B)) = \alpha((Aq_i a_j B)_T').$$

Заметим, что

$$\exp(c(i, j), n(T)) = 2^s \cdot 3^t \cdot 5^{n(D)}.$$

Поэтому при $i > 0$ функцию $S_2(n, i, j, u, v)$ можно задать равенством

$$S_2(n, i, j, u, v) = S_1(i, j, u, v, \exp(0, \exp(c(i, j), n)), \exp(1, \exp(c(i, j), n)), \exp(2, \exp(c(i, j), n))).$$

Остается дополнить это определение равенством

$$S_2(n, 0, j, u, v) = 2^u \cdot 3^0 \cdot 5^j \cdot 7^v.$$

Построим примитивно рекурсивную функцию $S_3(n, m)$ такую, что если $n = n(T)$ – номер машины Тьюринга T , $m = \alpha(K_T)$ – номер конфигурации K_T , а $(K_T)'_T$ – конфигурация, полученная из конфигурации K_T в результате выполнения одной команды, то

$$S_3(n, m) = \alpha((K_T)'_T).$$

Из выше сказанного следует, функцию $S_3(n, m)$ можно определить равенством

$$S_3(n, m) = S_2(n, \exp(1, m), \exp(2, m), \exp(0, m), \exp(3, m)).$$

Построим примитивно рекурсивную функцию $S(n, m, k)$ такую, что если $n = n(T)$ – номер машины Тьюринга T , $m = \alpha(K_T)$ – номер конфигурации K_T , а $(K_T)^{(k)}_T$ – конфигурация, полученная из конфигурации K_T за k тактов работы машины Тьюринга T , то

$$S(n, m, k) = \alpha((K_T)^{(k)}_T).$$

Функцию $S(n, m, k)$ можно определить схемой примитивной рекурсии

$$\begin{cases} S(n, m, 0) = m, \\ S(n, m, k+1) = S_3(n, S(n, m, k)) \end{cases}$$

Построим примитивно рекурсивную функцию pr_1 , которая по номеру $m = \alpha(K_T)$ конфигурации K_T дает число вхождений символа a_1 в конфигурацию K_T . Функцию pr_1 можно задать равенством

$$pr_1(m) = \sum_{i=0}^m \overline{sg}(|\exp(i, \exp(0, x)) - 1|) + \overline{sg}(|\exp(2, x) - 1|) + \sum_{i=0}^m \overline{sg}(|\exp(i, \exp(3, x)) - 1|).$$

Из определений построенных функций сразу следует, что если n -местная функция $f(x_1, \dots, x_n)$ вычислима на машине Тьюринга T с номером $n_0 = n(T)$, то

функция $f(x_1, \dots, x_n)$ определена на наборе (x_1, \dots, x_n) натуральных чисел тогда и только тогда, когда существует такое k , что

$$\exp(1, S(n_0, \delta_n(x_1, \dots, x_n), k)) = 0.$$

Рассмотрим частично рекурсивную функцию

$$\eta_n(t, x_1, \dots, x_n) = \mu_y(\exp(1, S(t, \delta_n(x_1, \dots, x_n), y)) = 0).$$

Теорема 2.1. Любая вычислимая по Тьюрингу функция частично рекурсивна.

Д о к а з а т е л ь с т в о. Если n -местная функция $f(x_1, \dots, x_n)$ вычислима на машине Тьюринга T с номером $n_0 = n(T)$, то выполняется равенство

$$f(x_1, \dots, x_n) = pr_1(S(n_0, \delta_n(x_1, \dots, x_n), \eta_n(n_0, x_1, \dots, x_n))).$$

□

Следствие. Если функция вычислима по Тьюрингу, то она правильно вычислима на некоторой машине Тьюринга с внешним алфавитом, состоящим лишь из двух символов.

Д о к а з а т е л ь с т в о. Если функция вычислима по Тьюрингу, то она частично рекурсивна. А каждая частично рекурсивная функция правильно вычислима на некоторой машине Тьюринга с внешним алфавитом, состоящим лишь из двух символов. □

Рассмотрим вопрос о существовании универсальных функций для различных классов частично рекурсивных функций.

Определение 2.1. $n+1$ -местная функция $U(t, x_1, \dots, x_n)$ называется универсальной для класса K n -местных функций, если

- 1) при любом фиксированном t_0 n -местная функция $U(t_0, x_1, \dots, x_n)$ принадлежит классу K ,
- 2) для любой функции $f(x_1, \dots, x_n)$ из класса K найдется такое число t_0 , что n -местная функция $U(t_0, x_1, \dots, x_n)$ совпадает с функцией $f(x_1, \dots, x_n)$.

Теорема 2.2. При любом n существует $n+1$ -местная частично рекурсивная функция, универсальная для класса всех n -местных частично рекурсивных функций.

Д о к а з а т е л ь с т в о. Рассмотрим $n+1$ -местную частично рекурсивную функцию

$$U^{(n+1)}(t, x_1, \dots, x_n) = pr_1(S(t, \delta_n(x_1, \dots, x_n), \eta_n(t, x_1, \dots, x_n))).$$

Из выше сказанного следует, что $n+1$ -местная частично рекурсивная функция $U^{(n+1)}(t, x_1, \dots, x_n)$ является универсальной для класса всех n -местных частично рекурсивных функций. □

Теорема 2.3. *Не существует двуместной примитивно рекурсивной функции, универсальной для класса всех одноместных примитивно рекурсивных функций.*

Теорема 2.4. *Не существует двуместной рекурсивной функции универсальной для класса всех одноместных рекурсивных функций.*

Д о к а з а т е л ь с т в о. Доказательство этих двух теорем проводится по единой схеме.

Обозначим через K класс всех одноместных примитивно рекурсивных функций. Предположим, что для него существует универсальная двуместная примитивно рекурсивная функция $U(t, x)$. Рассмотрим функцию $D(x) = U(x, x) + 1$. Тогда $D(x) \in K$. Поэтому найдется такое число n_0 , что при любом x выполняется равенство

$$D(x) = U(n_0, x).$$

Это равенство при $x = n_0$ дает противоречие

$$U(n_0, n_0) + 1 = U(n_0, n_0).$$

Для рекурсивных функций доказательство проводится простой заменой слов "примитивно рекурсивная функция" на "рекурсивная функция". \square

Оказывается, что для класса всех одноместных примитивно рекурсивных функций существует универсальная двуместная рекурсивная функция. Так как нам этот факт в дальнейшем не потребуется, то мы не будем приводить соответствующее достаточно сложное доказательство.

Рассмотрим исторически важные предикаты, связанные с вычислимостью на машинах Тьюринга.

Первым из них будет предикат, который мы обозначим через $T(z, x, y, t)$.

Полагаем

$$T(z, x, y, t) \iff$$

z – номер машины Тьюринга T ,
которая переводит конфигурацию $q_1 0 1^x 0$
в конфигурацию вида $q_0 0 1^y 0 \dots 0$ не более чем за t шагов.

Выше была построена примитивно рекурсивная функция $S(n, m, k)$ такая, что если $n = n(T)$ – номер машины Тьюринга T , $m = \alpha(K_T)$ – номер конфигурации K_T , а $(K_T)_T^{(k)}$ – конфигурация, полученная из конфигурации K_T за k тактов работы машины Тьюринга T , то

$$S(n, m, k) = \alpha((K_T)_T^{(k)}).$$

Примитивная рекурсивность предиката $T(z, x, y, t)$ следует из эквивалентности

$$T(z, x, y, t) \iff Tm(z) \& S(z, \delta_1(x), t) = 2^0 3^0 \cdot 5^0 7^{\prod_{i=1}^y p_{i-1}}.$$

Обозначим через $T_1(z, x, y)$ предикат $T(z, x, l(y), r(y))$. Тогда *частично рекурсивная функция*

$$U(z, x) = l(\mu_y(T_1(z, x, y)))$$

является универсальной для класса всех одноместных частично рекурсивных функций.

Аналогично для любого n n -местная частично рекурсивная функция

$$U(z, x_1, \dots, x_n) = l(\mu_y(T_1(z, c_n(x_1, \dots, x_n), y)))$$

является универсальной для класса всех n -местных частично рекурсивных функций.

Обозначим через $T_n(z, x_1, \dots, x_n, y)$ предикат $T_1(z, c_n(x_1, \dots, x_n), y)$, а через $\chi_{T_n}(z, x_1, \dots, x_n, y)$ – его характеристическую функцию. Тогда любая n -местная частично рекурсивная функция $f(x_1, \dots, x_n)$ представима в виде

$$f(x_1, \dots, x_n) = l(\mu_y(\overline{sg}(\chi_{T_n}(z_0, x_1, \dots, x_n, y)) = 0))$$

при некотором фиксированном z_0 . Последнее представление носит название *нормальная форма Клини*.

Рассмотрим некоторые алгоритмические проблемы для машин Тьюринга.

Проблема применимости. Разработать алгоритм, позволяющий по произвольной машине Тьюринга T и по произвольной конфигурации K_T определить, остановится ли машина T , начав работать в этой конфигурации K_T .

Проблема самоприменимости. Разработать алгоритм, позволяющий по произвольной машине Тьюринга T определить, остановится ли машина T , начав работать в конфигурации $q_1 01^{n(T)} 0$.

Проблема применимости к пустой ленте. Разработать алгоритм, позволяющий по произвольной машине Тьюринга T определить, остановится ли машина T , начав работать в конфигурации $q_1 0$.

Теорема 2.5. Проблемы применимости, самоприменимости и применимости к пустой ленте алгоритмически неразрешимы.

До к а з а т е л ь с т в о. Докажем алгоритмическую неразрешимость проблемы самоприменимости.

Обозначим через M_0 множество номеров всех самоприменимых машин Тьюринга, а через χ_{M_0} – его характеристическую функцию. В силу **тезиса Черча** необходимо доказать нерекурсивность функции χ_{M_0} .

Предположим противное, т.е. что функция χ_{M_0} рекурсивна. Тогда она правильно вычислима на некоторой машине Тьюринга T_0 с внешним алфавитом $A_{T_0} = \{0, 1\}$.

Рассмотрим вспомогательную машину Тьюринга V с внешним алфавитом $A_V = \{0, 1\}$, с внутренним алфавитом $Q_V = \{q_0, q_1, q_2\}$ и с программой, состоящей из трех команд

$$q_1 0 \quad \Rightarrow \quad q_2 0 R, \quad q_2 0 \quad \Rightarrow \quad q_0 0 S, \quad q_2 1 \quad \Rightarrow \quad q_2 1 S.$$

Обозначим через T_1 композицию $T_0 \cdot V$ машин Тьюринга T_0 и V , а через $n_1 = n(T_1)$ – ее номер.

Если $n_1 \in M_0$, т.е. машина Тьюринга T_1 *самоприменима*, то, начав работать в конфигурации $q_1 01^{n_1} 0$, она остановится, т.е. перейдет в конфигурацию вида $Cq_0 D$. С другой стороны, если машина T_1 начинает работать в конфигурации $q_1 01^{n_1} 0$, то сначала работает машина T_0 и переводит эту конфигурацию в конфигурацию вида $q_0 010 \dots 0$ (машина Тьюринга T_1 *самоприменима*). Затем начнет работать машина V , которая конфигурацию $q_1 010 \dots 0$ переведет в конфигурацию $0q_2 10 \dots 0$ и будет оставаться в ней "вечно", в частности, не перейдет в конфигурацию вида $Cq_0 D$. Полученное противоречие доказывает, что $n_1 \notin M_0$, т.е. машина Тьюринга T_1 *несамоприменима*. Значит, начав работать в конфигурации $q_1 01^{n_1} 0$, она не остановится, т.е. не перейдет в конфигурацию вида $Cq_0 D$. С другой стороны, если машина T_1 начинает работать в конфигурации $q_1 01^{n_1} 0$, то сначала работает машина T_0 и переводит эту конфигурацию в конфигурацию вида $q_0 00 \dots 0$ (машина Тьюринга T_1 *несамоприменима*). Затем начнет работать машина V , которая конфигурацию $q_1 00 \dots 0$ переведет последовательно в конфигурацию $0q_2 0 \dots 0$, $0q_0 0 \dots 0$, т.е. остановится.

Полученное противоречие доказывает, что наше предположение о рекурсивности функции χ_{M_0} ведет к противоречию, значит *функция χ_{M_0} не является рекурсивной*.

Неразрешимость проблемы применимости для машин Тьюринга, конечно, следует из неразрешимости для них проблемы самоприменимости. \square

В заключение построим *рекурсивно перечислимое, но не рекурсивное множество* и *машину Тьюринга, для которой неразрешима проблема применимости*.

Выше была установлена примитивная рекурсивность предиката

$$T(z, x, y, t) \iff$$

$$z - \text{номер машины Тьюринга } T,$$

$$\text{которая переводит конфигурацию } q_1 01^x 0$$

$$\text{в конфигурацию вида } q_0 01^y 0 \dots 0 \text{ не более чем за } t \text{ шагов.}$$

Через $T_1(z, x, y)$ был обозначен предикат $T(z, x, l(y), r(y))$ и доказано, что *частично рекурсивная функция*

$$U(z, x) = l(\mu_y(T_1(z, x, y)))$$

является универсальной для класса всех одноместных частично рекурсивных функций.

Рассмотрим рекурсивно перечислимое множество

$$C = \{ x \mid (\exists y) T_1(x, x, y) \}.$$

Нетрудно понять, что C – *множество номеров всех самоприменимых машин Тьюринга*. Покажем, что C – *рекурсивно перечислимое, но не рекурсивное множество*.

Предположим, что множество C рекурсивно. Тогда его дополнение рекурсивно, поэтому частично рекурсивна и частичная характеристическая функция $\chi_{N \setminus C}^{(c)}(x)$. Найдется такое число n , что

$$\chi_{N \setminus C}^{(c)}(x) = U(n, x) = l(\mu_y(T_1(n, x, y))).$$

Предположим, что $n \in C$. Тогда значение

$$\chi_{N \setminus C}^{(c)}(n) = U(n, n) = l(\mu_y(T_1(n, n, y)))$$

не определено. Так как предикат $T_1(n, n, y)$ примитивно рекурсивен, то это возможно лишь в случае, когда $\neg(\exists y)T_1(n, n, y)$, а значит, $n \notin C$. Полученное противоречие доказывает, что $n \notin C$. Значит $\neg(\exists y)T_1(n, n, y)$, поэтому $l(\mu_y(T_1(n, n, y)))$ не определено. Но тогда не определено и $\chi_{N \setminus C}^{(c)}(n)$, значит, $n \in C$.

Полученное противоречие показывает, что сделанное предположение о рекурсивности множества C неверно.

Так как множество C рекурсивно перечислимо, то его частичная характеристическая функция $\chi_C^{(c)}(x)$ частично рекурсивна, а значит, существует машина Тьюринга T_c с внешним алфавитом $\{0, 1\}$, правильно ее вычисляющая. Т.е. если $n \in C$, то машина Тьюринга T_c , начав работать в конфигурации $q_1 01^n 0$, через конечное число тактов работы остановится (в конфигурации вида $q_0 010 \dots 0$); если же $n \notin C$, то машина Тьюринга T_c , начав работать в конфигурации $q_1 01^n 0$, никогда не остановится.

Поэтому машина Тьюринга T_c применима к конфигурации $q_1 01^n 0$ тогда и только тогда, когда $n \in C$.

Так как множество C не является рекурсивным, то в силу тезиса Черча **проблема применимости для машины Тьюринга T_c алгоритмически неразрешима.**

Выше нами была использована частичная характеристическая функция $\chi_C^{(c)}(x)$ множества C , являющаяся частично рекурсивной. Покажем, что наличие подобных не всюду определенных частично рекурсивных функций обусловлено достаточно глубокими причинами. Более точно, покажем, что *существует частично рекурсивная функция, которую нельзя доопределить до всюду определенной рекурсивной функции.*

Пусть $U(n, x)$ – двуместная частично рекурсивная функция, универсальная для класса всех одноместных частично рекурсивных функций. Рассмотрим функцию $V(x) = \overline{sg}U(x, x)$. Предположим, что она имеет рекурсивное доопределение, т.е. существует рекурсивная функция $f(x)$ такая, что если в точке n значение функции $V(x)$ определено, то $f(n) = V(n)$.

Так как двуместная частично рекурсивная функция $U(n, x)$ является универсальной для класса всех одноместных частично рекурсивных функций, то найдется такое число n , что для всех x выполнено равенство

$$U(n, x) = f(x).$$

При $x = n$ получаем противоречие

$$U(n, n) = f(n) = V(n) = \overline{sg}U(n, n).$$

Свяжем с функцией $V(x)$ два множества

$$V_0 = \{x \mid V(x, x) = 0\}, \quad V_1 = \{x \mid V(x, x) = 1\}.$$

Множества V_0 и V_1 рекурсивно перечислимы и не пересекаются.

Покажем, что множества V_0 и V_1 рекурсивно неотделимы, т.е. не существует двух непересекающихся рекурсивных множеств A и B таких, что $V_0 \subseteq A$ и $V_1 \subseteq B$. Допустим, что такие рекурсивные множества A и B существуют. Тогда $V_1 \subseteq N \setminus A$ и характеристическая функция $\chi_{N \setminus A}(x)$ множества $N \setminus A$ является рекурсивным доопределением функции $V(x)$, что, как показано выше, невозможно.

§3. Нумерация функций и множеств

Частично рекурсивная функция $U(z, x)$, универсальная для класса всех одноместных частично рекурсивных функций, позволяет каждой одноместной частично рекурсивной функции $f(x)$ сопоставить в качестве ее номера натуральное число n такое, что для любого x выполняется равенство

$$f(x) = U(n, x).$$

Тот факт, что у функции может быть не один номер, не играет существенной роли.

Аналогичным образом, используя $n + 1$ -местную частично рекурсивную функцию $U(z, x_1, \dots, x_n)$, являющуюся универсальной для класса всех n -местных частично рекурсивных функций, можно каждой n -местной частично рекурсивной функции сопоставить ее номер.

Так как любое рекурсивно перечислимое множество есть множество значений некоторой одноместной частично рекурсивной функции, можно занумеровать и рекурсивно перечислимые множества.

Рассмотрим так называемую нумерацию Клини частично рекурсивных функций и нумерацию Поста рекурсивно перечислимых множеств.

Будем использовать стандартные в этой теории обозначения

$$[x, y] = c(l(x), c(r(x), y)),$$

$$[x]_{21} = c(l(x), l(r(x))), \quad [x]_{22} = r(r(x)).$$

Легко проверить, что выполняются равенства

$$[[x, y]]_{21} = x, \quad [[x, y]]_{22} = y, \quad [[x]_{21}, [x]_{22}] = x.$$

Значит $[x, y]$, $[x]_{21}$ и $[x]_{22}$ — нумерационные функции для пар натуральных чисел. Их определенные преимущества перед канторовскими нумерационными функциями $c(x, y)$, $l(x)$ и $r(x)$ будут видны позже.

Далее стандартным приемом индукцией по n определяются нумерационные функции для наборов длины n натуральных чисел

$$\begin{aligned} [x_1, x_2, x_3, \dots, x_n] &= [[x_1, x_2, x_3, \dots, x_{n-1}], x_n], \\ [x]_{n1} &= [[x]_{21}]_{n-1,1}, [x]_{n2} = [[x]_{21}]_{n-1,2}, \\ &\dots \\ [x]_{n,n-1} &= [[x]_{21}]_{n-1,n-1}, [x]_{n,n} = [x]_{22}. \end{aligned}$$

Индукцией по n устанавливается справедливость равенств

$$[[x_1, x_2, x_3, \dots, x_n]]_{ni} = x_i, \quad [[x]_{n1}, [x]_{n2}, \dots, [x]_{nn}] = x.$$

В дальнейшем часто будет использоваться равенство

$$[x_1, x_2, x_3, \dots, x_m, x_{m+1}, \dots, x_n] = [[x_1, x_2, x_3, \dots, x_m], x_{m+1}, \dots, x_n],$$

справедливость которого сразу следует из определения функции

$$[x_1, x_2, x_3, \dots, x_n].$$

Кроме того, имеет место своеобразная "ассоциативность"

$$c(x_1, c(x_2, x_3)) = [c(x_1, x_2), x_3].$$

Используя построенную выше частично рекурсивную функцию $U(z, x)$, универсальную для класса всех одноместных частично рекурсивных функций, для любого n строится $n + 1$ -местная функция $K^{n+1}(x_0, x_1, \dots, x_n)$, универсальная для класса всех n -местных частично рекурсивных функций

$$\begin{aligned} K^2(x_0, x_1) &= U(l(x_0), c(r(x_0), x_1)), \\ K^{n+1}(x_0, x_1, x_2, \dots, x_n) &= K^n([x_0, x_1], x_2, \dots, x_n). \end{aligned}$$

$n + 1$ -местная функция $K^{n+1}(x_0, x_1, \dots, x_n)$ называется *клинисевской нумерационной функцией* для класса всех n -местных частично рекурсивных функций.

Сразу из определений следуют равенства

$$\begin{aligned} K^{n+1}(x_0, x_1, x_2, \dots, x_n) &= K^2([\dots [[x_0, x_1], x_2], \dots, x_{n-1}], x_n) = \\ &= K^2([x_0, x_1, x_2, \dots, x_{n-1}], x_n). \end{aligned}$$

Поэтому, используя равенство

$$[x_1, x_2, x_3, \dots, x_m, x_{m+1}, \dots, x_n] = [[x_1, x_2, x_3, \dots, x_m], x_{m+1}, \dots, x_n],$$

получаем, что для любых n и m справедливо равенство

$$K^{n+m+1}(x_0, x_1, x_2, x_3, \dots, x_m, x_{m+1}, \dots, x_{m+n}) = K^{n+1}([x_0, x_1, x_2, x_3, \dots, x_m], x_{m+1}, \dots, x_{m+n}).$$

Покажем, что для любого n выполняется равенство

$$U^{n+1}(x_0, x_1, x_2, \dots, x_n) = K^n(c(x_0, x_1), x_2, \dots, x_n).$$

Напомним, что

$$U^{n+1}(x_0, x_1, x_2, \dots, x_n) = U(x_0, c_n(x_1, x_2, \dots, x_n)).$$

При $n = 2$ получаем

$$K^2(c(x_0, x_1), x_2) = U(x_0, c(x_1, x_2)) = U^3(x_0, x_1, x_2).$$

Предположив выполненным равенство

$$U^{n+1}(x_0, x_1, x_2, \dots, x_n) = K^n(c(x_0, x_1), x_2, \dots, x_n),$$

получим

$$\begin{aligned} K^{n+1}(c(x_0, x_1), x_2, x_3, \dots, x_{n+1}) &= K^n([c(x_0, x_1), x_2], x_3, \dots, x_{n+1}) = \\ &= K^n(c(x_0, c(x_1, x_2)), x_3, \dots, x_{n+1}) = U^{n+1}(x_0, c(x_1, x_2), x_3, \dots, x_n) = \\ &= U^{n+2}(x_0, x_1, x_2, x_3, \dots, x_n), \end{aligned}$$

так как $c_{n-1}(c(x_1, x_2), x_3, \dots, x_n) = c_n(x_1, x_2, x_3, \dots, x_n)$.

Покажем, что для любой n -местной частично рекурсивной функции $f(x_1, \dots, x_n)$ найдется такое число x_0 , что для всех x_1, \dots, x_n выполняется равенство

$$f(x_1, \dots, x_n) = K^{n+1}(x_0, x_1, x_2, \dots, x_n).$$

Если функцию $f(x_1, \dots, x_n)$ рассматривать как функцию от переменных u, x_1, \dots, x_n , то найдется такое число b , что

$$f(x_1, \dots, x_n) = U^{n+2}(b, u, x_1, \dots, x_n) = K^{n+1}(c(b, u), x_1, \dots, x_n).$$

Если у частично рекурсивной $m + n$ -местной функции

$$f(y_1, \dots, y_m, x_1, \dots, x_n)$$

зафиксировать первые m аргументов, получим частично рекурсивную n -местную функцию $f_{y_1, \dots, y_m}(x_1, \dots, x_n)$. Равенство

$$K^{n+m+1}(x_0, y_1, y_2, y_3, \dots, y_m, x_1, \dots, x_n) = K^{n+1}([x_0, y_1, y_2, y_3, \dots, y_m], x_1, \dots, x_n)$$

показывает, что *примитивно рекурсивная функция*

$$[x_0, y_1, y_2, y_3, \dots, y_m]$$

по номеру x_0 частично рекурсивной $m + n$ -местной функции

$$f(y_1, \dots, y_m, x_1, \dots, x_n)$$

и по фиксированным значениям y_1, y_2, \dots, y_m ее первых m аргументов дает номер частично рекурсивной n -местной функции

$$f_{y_1, \dots, y_m}(x_1, \dots, x_n).$$

Аналогичным образом доказывается, что *существуют примитивно рекурсивные функции, которые по номерам функций дают номера функций, получающихся из них суперпозицией, примитивной рекурсией или минимизацией*. Ради некоторого сокращения будем использовать обозначение \bar{x} для набора x_1, \dots, x_n .

Пусть n -местная функция f получена суперпозицией из m -местной частично рекурсивной функции g и n -местных частично рекурсивных функций f_1, \dots, f_m . Построим *примитивно рекурсивную функцию, которая по номерам функций g, f_1, \dots, f_m дает номер функции f* .

Рассмотрим $n + m + 1$ -местную функцию

$$K^{m+1}(z, K^{n+1}(y_1, \bar{x}), \dots, K^{n+1}(y_m, \bar{x})).$$

Найдется такое число s , что выполняются равенства

$$\begin{aligned} K^{m+1}(z, K^{n+1}(y_1, \bar{x}), \dots, K^{n+1}(y_m, \bar{x})) = \\ K^{n+m+2}(s, z, y_1, \dots, y_m, \bar{x}) = K^{n+1}([s, z, y_1, \dots, y_m], \bar{x}). \end{aligned}$$

В качестве искомой функции можно взять $[s, z, y_1, \dots, y_m]$, так как если z – номер функции g , y_1 – номер функции f_1 , \dots , y_m – номер функции f_m , то $[s, z, y_1, \dots, y_m]$ – номер функции $f = S(g; f_1, \dots, f_m)$.

Пусть $n + 1$ -местная функция f получена примитивной рекурсией из n -местной частично рекурсивной функции g и $n + 2$ -местной частично рекурсивной функции h . Построим примитивно рекурсивную функцию, которая по номерам функций g и h дает номер функции f . Зададим схемой примитивной рекурсии $n + 3$ -местную функцию

$$\begin{cases} F(u, v, \bar{x}, 0) = K^{n+1}(u, \bar{x}), \\ F(u, v, \bar{x}, y + 1) = K^{n+3}(v, y, \bar{x}, F(u, v, \bar{x}, y)). \end{cases}$$

Так как $n + 3$ -местная функция $F(u, v, \bar{x}, y)$ частично рекурсивна, то найдется такое натуральное число pr , что выполнены равенства

$$F(u, v, \bar{x}, y) = K^{n+4}(pr, u, v, \bar{x}, y) = K^{n+2}([pr, u, v], \bar{x}, y).$$

Примитивно рекурсивная функция $[pr, u, v]$ по номеру u функций g и номеру v функции h дает номер функции $f = PR(g; h)$.

Пусть n -местная функция f получена оператором минимизации из $n + 1$ -местной частично рекурсивной функции g . Построим примитивно рекурсивную функцию, которая по номеру функции g дает номер функции f .

Зададим оператором минимизации $n + 1$ -местную функцию

$$F(u, \bar{x}) = \mu_y(K^{n+1}(u, \bar{x}, y) = 0).$$

Так как $n + 1$ -местная функция $F(u, \bar{x})$ частично рекурсивна, то найдется такое натуральное число μ , что выполнены равенства

$$F(u, \bar{x}) = K^{n+2}(\mu, u, \bar{x}) = K^{n+1}([\mu, u], \bar{x}).$$

Примитивно рекурсивная функция $[\mu, u]$ по номеру u функций g дает номер функции $f = \mu_y(g)$.

Рассмотрим некоторые вопросы, связанные с нумерацией одноместных функций и рекурсивно перечислимых множеств.

Если для одноместной функции $f(x)$ и числа n при любом x выполнено равенство $f(x) = K^2(n, x)$, то число n называется **клиниевским номером функции** $f(x)$. Саму функцию $f(x)$ в этом случае будем обозначать через κ_n .

Обозначим через ω *нигде не определенную функцию*.

Покажем, что для любой частично рекурсивной функции $f(x_1, \dots, x_n)$ существует такая примитивно рекурсивная функция $g(x_1, \dots, x_n)$, что если на наборе (a_1, \dots, a_n) натуральных чисел функция f определена, то

$$\kappa_{g(a_1, \dots, a_n)} = \kappa_{f(a_1, \dots, a_n)}.$$

В противном случае $\kappa_{g(a_1, \dots, a_n)} = \omega$.

Для доказательства рассмотрим функцию $K^2(f(x_1, \dots, x_n), y)$. Найдется такое число a , что

$$K^2(f(x_1, \dots, x_n), y) = K^{n+2}(a, x_1, \dots, x_n, y) = K^2([a, x_1, \dots, x_n], y)$$

Примитивно рекурсивную функцию $[a, x_1, \dots, x_n]$ можно взять в качестве $g(x_1, \dots, x_n)$.

Теорема 3.1 (Теорема о неподвижной точке). *Для произвольной частично рекурсивной функции $g(x_1, \dots, x_n, y)$ существует такая примитивно рекурсивная функция $f(x_1, \dots, x_n)$, что если на наборе (a_1, \dots, a_n) натуральных чисел значение функции*

$$g(x_1, \dots, x_n, f(x_1, \dots, x_n))$$

определено, то

$$\kappa_{g(a_1, \dots, a_n, f(a_1, \dots, a_n))} = \kappa_{f(a_1, \dots, a_n)}.$$

В противном случае $\kappa_{g(a_1, \dots, a_n)}$ – нигде не определенная функция ω .

Д о к а з а т е л ь с т в о. Рассмотрим функцию

$$K^{n+3}(f(x_1, \dots, x_n), [u, u, x_1, \dots, x_n], y).$$

Найдется такое число e , что выполнены равенства

$$K^2(f(x_1, \dots, x_n), [u, u, x_1, \dots, x_n], y) = \\ K^{n+3}(e, u, x_1, \dots, x_n, y) = K^2([e, u, x_1, \dots, x_n], y).$$

В качестве искомой функции $f(x_1, \dots, x_n)$ можно взять $[e, e, x_1, \dots, x_n]$. \square

При $n = 0$ получаем

Следствие. Для произвольной частично рекурсивной функции $g(y)$ существует такое число e , что если значение функции $g(e)$ определено, то

$$\kappa_{g(e)} = \kappa_e.$$

В противном случае κ_e – нигде не определенная функция ω .

Докажем одну из фундаментальных теорем в теории частично рекурсивных функций – теорему Райса, в соответствии с которой *никакое нетривиальное свойство частично рекурсивных функций нераспознаваемо по их клиниевским номерам.*

Теорема 3.2 (Теорема Райса). Для любого непустого класса K одноместных частично рекурсивных функций, отличного от класса всех одноместных частично рекурсивных функций, множество

$$U = \{ n \mid \kappa_n \in K \}$$

не является рекурсивным.

Д о к а з а т е л ь с т в о. Предположим, что множество U рекурсивно. Тогда рекурсивно и его дополнение. Пусть $a \in U$, $b \in N \setminus U$. Рассмотрим рекурсивную функцию

$$f(x) = a \cdot \chi_{N \setminus U}(x) + b \cdot \chi_U(x).$$

По предыдущему следствию найдется такое число e , что $g = \kappa_e = \kappa_{f(e)}$.

Если $e \in U$, то $g = \kappa_e \in K$. Однако в этом случае $f(e) = b \notin U$. Значит, $g = \kappa_{f(e)} \notin K$. Полученное противоречие показывает, что $e \notin U$. Тогда $g = \kappa_e \notin K$. Однако в этом случае $f(e) = a \in U$. Значит $g = \kappa_{f(e)} \in K$.

Таким образом предположение о рекурсивности множества U ведет к противоречию. \square

Нумерация частично рекурсивных функций дает *нумерацию рекурсивно перечислимых множеств.*

Если число n таково, что рекурсивно перечислимое множество U совпадает с множеством значений частично рекурсивной функции $K^2(n, x)$, то число n называется **постовским номером множества U** , которое в этом случае обозначается через π_n .

Для рекурсивно перечислимых множеств также справедлива теорема Райса, в соответствии с которой *никакое нетривиальное свойство рекурсивно перечислимых множеств нераспознаваемо по их постовским номерам*.

Теорема 3.3 (Теорема Райса). *Для любого непустого класса K рекурсивно перечислимых множеств, отличного от класса всех рекурсивно перечислимых множеств, множество*

$$U = \{n \mid \pi_n \in K\}$$

не является рекурсивным.

Доказательство. Предположим, что множество U рекурсивно. Тогда рекурсивно и его дополнение. Пусть $a \in U$, $b \in N \setminus U$. Рассмотрим рекурсивную функцию

$$f(x) = a \cdot \chi_{N \setminus U}(x) + b \cdot \chi_U(x).$$

По предыдущему следствию найдется такое число e , что $g = \kappa_e = \kappa_{f(e)}$. Тогда $\pi = \pi_e = \pi_{f(e)}$.

Если $e \in U$, то $\pi = \pi_e \in K$. Однако в этом случае $f(e) = b \notin U$. Значит $\pi = \pi_{f(e)} \notin K$. Полученное противоречие показывает, что $e \notin U$. Тогда $\pi = \pi_e \notin K$. Однако в этом случае $f(e) = a \in U$. Значит, $\pi = \pi_{f(e)} \in K$.

Таким образом, предположение о рекурсивности множества U ведет к противоречию. \square

В качестве следствия теоремы о неподвижной точке 3.1 для частично рекурсивных множеств получаем следующую теорему.

Теорема 3.4 (Теорема о неподвижной точке). *Для произвольной частично рекурсивной функции $g(x_1, \dots, x_n, y)$ существует такая примитивно рекурсивная функция $f(x_1, \dots, x_n)$, что если на наборе (a_1, \dots, a_n) натуральных чисел значение функции*

$$g(x_1, \dots, x_n, f(x_1, \dots, x_n))$$

определено, то

$$\pi_{g(a_1, \dots, a_n, f(a_1, \dots, a_n))} = \pi_{f(a_1, \dots, a_n)}.$$

В противном случае $\pi_{g(a_1, \dots, a_n)}$ – пустое множество \emptyset .

При $n = 0$ получаем

Следствие. *Для произвольной частично рекурсивной функции $g(y)$ существует такое число e , что если значение функции $g(e)$ определено, то*

$$\pi_{g(e)} = \pi_e.$$

В противном случае π_e – пустое множество \emptyset .

Завершим параграф обсуждением чрезвычайно важного понятия – понятия *m -сводимости*. Рассмотрение более общего понятия сводимости – сводимости по Тьюрингу – выходит за задуманные рамки пособия. Однако позже мы рассмотрим более узкое понятие сводимости – полиномиальную сводимость.

Определение 3.1. *Подмножество A множества натуральных чисел m -сводимо к подмножеству B , если существует такая рекурсивная одноместная функция f , что для любого натурального числа a справедлива эквивалентность*

$$a \in A \iff f(a) \in B.$$

Про функцию f говорят, что она сводит подмножество A к подмножеству B .

Запись $A \leq_m B$ служит сокращением для утверждения

”подмножество A множества натуральных чисел m -сводимо к подмножеству B ”.

На самом деле следовало бы говорить о *сводимости проблемы вхождения во множество A к проблеме вхождения во множество B* . В этом случае про функцию f говорят, что она сводит проблему вхождения во множество A к проблеме вхождения во множество B . Однако мы будем придерживаться устоявшейся в этой области теории алгоритмов терминологии. А намеченная терминология будет использоваться в параграфах, посвященных сложности алгоритмов.

Фундаментальным фактом теории рекурсивно перечислимых множеств является теорема о существовании m -универсального рекурсивно перечислимого множества, т.е. такого рекурсивно перечислимого множества H , к которому m -сводимо любое рекурсивно перечислимое множество A .

Приведем пример m -универсального рекурсивно перечислимого множества. Полагаем

$$H = \{c(x, y) \mid x \in \pi_y\}.$$

Рекурсивная перечислимость множества H следует из эквивалентности

$$u \in H \iff (\exists x, y, t)(z = c(x, y) \& x = K^2(y, t)).$$

Установим m -универсальность множества H .

Пусть A – произвольное рекурсивно перечислимое множество, а e – его постовский номер, т.е. $A = \pi_e$. Тогда верна эквивалентность

$$u \in A = \pi_e \iff c(u, e) \in H.$$

Поэтому в качестве функции f , m -сводящей рекурсивно перечислимое множество $A = \pi_e$ к множеству H , можно взять функцию $f(x) = c(x, e)$.

Напомним, что по теореме Э. Поста *рекурсивно перечислимое множество A натуральных чисел является рекурсивным тогда и только тогда, когда его дополнение $N \setminus A$ рекурсивно перечисливо*.

Поэтому рекурсивно перечислимое множество A натуральных чисел не является рекурсивным тогда и только тогда, когда его дополнение $N \setminus A$ не является рекурсивно перечислимым. Последнее означает, что для любого рекурсивно перечислимого подмножества B множества $N \setminus A$ ($B \subseteq N \setminus A$) найдется такой элемент b , что $b \in N \setminus A$, но $b \notin B$. Про элемент b можно было бы сказать, что он устанавливает несовпадение рекурсивно перечислимого множества B с $N \setminus A$.

Это приводит к следующему определению.

Определение 3.2. Подмножество P множества натуральных чисел называется **продуктивным**, если существует такая рекурсивная функция $f(x)$, что для любого натурального числа n :

если $\pi_n \subseteq P$, то $f(n) \in P$ и $f(n) \notin \pi_n$.

Таким образом, функция $f(n)$ "эффективно" устанавливает несовпадение продуктивного множества P с его рекурсивно перечислимым подмножеством π_n . Отметим, что для продуктивного множества P верна импликация

$$\pi_n \subseteq P \implies f(n) \in P \setminus \pi_n.$$

Продуктивные множества, если они существуют, не являются рекурсивно перечислимыми.

Определение 3.3. Рекурсивно перечислимое подмножество K множества натуральных чисел называется **креативным**, если его дополнение $N \setminus K$ продуктивно.

Можно доказать, что подмножество K множества натуральных чисел N является креативным тогда и только тогда, когда оно ***m***-универсально. Однако в наши планы не входит столь фундаментальное изучение рекурсивно перечислимых множеств. За дальнейшими более глубокими сведениями о рекурсивно перечислимых множествах мы отсылаем заинтересованного читателя к обширной литературе по этой тематике.

ГЛАВА IV

АЛГОРИТМИЧЕСКИЕ ПРОБЛЕМЫ

§1. Арифметические множества и теорема А. Тарского

Как и ранее, будем обозначать через N множество $\{0, 1, 2, \dots, \dots\}$ натуральных чисел, включающее в себя по чисто техническим причинам нуль 0. На множестве N определены операции сложения $+$ и умножения \cdot . Рассмотрим сигнатуру

$$\tau_{Arithm} = \langle \{\bar{0}, \bar{1}\}, \{\bar{+}, \bar{\cdot}\}, \{\bar{=}\} \rangle,$$

где $\bar{0}$ и $\bar{1}$ – константные символы, $\bar{+}$ и $\bar{\cdot}$ – 2-местные функциональные символы, а $\bar{=}$ – 2-местный предикатный символ.

Язык $L_{\tau_{Arithm}}$ первого порядка сигнатуры τ_{Arithm} называется **языком арифметики**, а формулы этого языка – **арифметическими формулами**.

Индуктивно для каждого натурального числа n определим терм \bar{n} языка $L_{\tau_{Arithm}}$, полагая $\bar{n} + \bar{1} = (\bar{n} \bar{+} \bar{1})$.

Ограничимся рассмотрением лишь *стандартной интерпретации* I языка $L_{\tau_{Arithm}}$, для которой основным множеством служит множество N натуральных чисел, $I(\bar{0})$ – это 0, $I(\bar{1})$ – 1, $I(\bar{+})$ – $+$ (сложение натуральных чисел), $I(\bar{\cdot})$ – \cdot (умножение натуральных чисел), а $I(\bar{=})$ – обычное отношение равенства натуральных чисел. Обычным образом определяется *значение замкнутого терма* и *истинностное значение замкнутой формулы в стандартной интерпретации*. Чтобы не усложнять обозначения в дальнейшем вместо $\bar{0}$, $\bar{1}$, $\bar{+}$, $\bar{\cdot}$, $\bar{=}$ и \bar{n} будем, как правило, просто писать 0, 1, +, ·, = и n , что не приведет к двусмысленности, так как из контекста будет ясно идет ли речь о натуральных числах или о соответствующих термах языка $L_{\tau_{Arithm}}$.

Введем три основных понятия этого параграфа – понятия **арифметического предиката**, **арифметического множества** и **арифметической функции**.

Определение 1.1. n -местный предикат $P(x_1, \dots, x_n)$ на множестве N натуральных чисел называется **арифметическим**, если существует такая арифметическая формула Φ_P со свободными переменными x_1, \dots, x_n , что для произвольных натуральных чисел a_1, \dots, a_n имеет место эквивалентность:

значение предиката $P(a_1, \dots, a_n)$ есть "истина" тогда и только тогда, когда формула $(\Phi_P)_{x_1, \dots, x_n} [\bar{a}_1, \dots, \bar{a}_n]$ истинна в стандартной интерпретации.

Определение 1.2. Множество U n -ок натуральных чисел, т.е. подмножество множества N^n , называется **арифметическим**, если существует такая арифметическая формула Φ_U со свободными переменными x_1, \dots, x_n , что для произвольных натуральных чисел a_1, \dots, a_n имеет место эквивалентность:

$(a_1, \dots, a_n) \in U$ тогда и только тогда, когда в стандартной интерпретации истинна формула $(\Phi_U)_{x_1, \dots, x_n} [\bar{a}_1, \dots, \bar{a}_n]$.

Определение 1.3. n -местная функция $y = f(x_1, \dots, x_n)$ называется **арифметической**, если существует такая арифметическая формула Φ_f со свободными переменными y, x_1, \dots, x_n , что для произвольных натуральных чисел b, a_1, \dots, a_n имеет место эквивалентность:

$b = f(a_1, \dots, a_n)$ тогда и только тогда, когда в стандартной интерпретации истинна формула $(\Phi_f)_{y, x_1, \dots, x_n} [\bar{b}, \bar{a}_1, \dots, \bar{a}_n]$.

Теорема 1.1. Каждая частично рекурсивная функция является арифметической.

Каждое рекурсивно перечислимое множество n -ок натуральных чисел является арифметическим.

Д о к а з а т е л ь с т в о. Для доказательства арифметичности любой частично рекурсивной функции достаточно установить арифметичность исходных (простейших) частично рекурсивных функций и замкнутость класса арифметических функций относительно операций суперпозиции, примитивной рекурсии и минимизации.

Арифметичность исходных частично рекурсивных функций: нулевой функции $y = 0(x_1)$, функции следования $y = s(x_1) = x_1 + 1$, функций проектирования $y = U_m^n(x_1, \dots, x_n) = x_m$ и постоянных 0-местных функций a , равных константе a очевидна.

Докажем, что операция суперпозиции, будучи примененной к арифметическим функциям, дает арифметическую функцию.

Пусть $g(x_1, \dots, x_m), f_1(x_1, \dots, x_n), \dots, f_m(x_1, \dots, x_n)$ – арифметические функции, $\Phi_g, \Phi_{f_1}, \dots, \Phi_{f_m}$ – соответствующие арифметические формулы, а функция $f(x_1, \dots, x_n)$ получена из этих функций операцией суперпозиции, т.е. задается равенством

$$f(a_1, \dots, a_n) = g(f_1(a_1, \dots, a_n), \dots, f_m(a_1, \dots, a_n)).$$

Тогда арифметичность функции $f(x_1, \dots, x_n)$ следует из эквивалентности

$$b = f(a_1, \dots, a_n) \iff (\exists y_1) \dots (\exists y_m)((\Phi_{f_1})_{y, x_1, \dots, x_n}[y_1, \overline{a_1}, \dots, \overline{a_n}] \& \dots \& (\Phi_{f_m})_{y, x_1, \dots, x_n}[y_m, \overline{a_1}, \dots, \overline{a_n}] \& (\Phi_g)_{y, x_1, \dots, x_m}[\overline{b}, y_1, \dots, y_m]).$$

Покажем, что оператор минимизации, будучи примененным к арифметическим функциям, дает арифметическую функцию.

Пусть n -местная функция f получается из арифметических $n + 1$ -местных функций g и h применением оператора минимизации, т.е. выполнено следующее условие:

для произвольных натуральных чисел a_1, \dots, a_n и b выполняется равенство $f(a_1, \dots, a_n) = b$ тогда и только тогда, когда при любом $t < b$ значения $g(a_1, \dots, a_n, t)$ и $h(a_1, \dots, a_n, t)$ определены и не равны, а значения $g(a_1, \dots, a_n, b)$ и $h(a_1, \dots, a_n, b)$ определены и равны.

Пусть арифметические формулы Φ_g и Φ_h обеспечивают арифметичность функций $y = g(x_1, \dots, x_n, t)$ и $y = h(x_1, \dots, x_n, t)$. Тогда арифметичность функции $y = f(x_1, \dots, x_n)$ следует из эквивалентности

$$b = f(a_1, \dots, a_n) \iff (\exists y)((\Phi_g)_{y, x_1, \dots, x_n, t}[y, \overline{a_1}, \dots, \overline{a_n}, \overline{b}] \& (\Phi_h)_{y, x_1, \dots, x_n, t}[y, \overline{a_1}, \dots, \overline{a_n}, \overline{b}]) \& (\forall z)(z < \overline{b} \leftarrow (\exists y_1)(\exists y_2)(\Phi_g)_{y, x_1, \dots, x_n, t}[y_1, \overline{a_1}, \dots, \overline{a_n}, z] \& (\Phi_h)_{y, x_1, \dots, x_n, t}[y_2, \overline{a_1}, \dots, \overline{a_n}, z] \& y_1 \neq y_2).$$

Для доказательства замкнутости класса арифметических функций относительно применения оператора примитивной рекурсии нам потребуется функция Геделя, осуществляющая нумерацию всех конечных последовательностей натуральных чисел.

Напомним, что ранее была построена примитивно рекурсивная функция $\Gamma(x, t) = \text{rest}(l(x), 1 + (t + 1)r(x))$ такая, что для любой конечной последовательности натуральных чисел a_0, a_1, \dots, a_n (произвольной длины n) найдется такое число m , что

$$\bigwedge_{i=0}^n \Gamma(m, i) = a_i.$$

Вместо функции $\Gamma(x, t)$ мы возьмем функцию от трех переменных

$$\text{rest}(x, 1 + (t + 1)z),$$

из которой и получалась функция $\Gamma(x, t)$. Как было установлено ранее, для любой конечной последовательности натуральных чисел a_0, a_1, \dots, a_n (произвольной длины n) найдутся такие числа c и d , что

$$\bigwedge_{i=0}^n \text{rest}(c, 1 + (i + 1)d) = a_i.$$

Функцию $rest(x, 1 + (t + 1)z)$ будем обозначать через $\Gamma_0(x, z, t)$. Ее арифметичность следует из эквивалентности

$$b = rest(a, 1 + (c + 1)d) \iff (\exists u)(\bar{a} = \overline{1 + (c + 1)d} \cdot \bar{u} + \bar{b} \ \& \ \bar{b} < \overline{1 + (c + 1)d}).$$

Через Φ_{Γ_0} обозначим следующую формулу

$$(\exists u)(x = (\bar{1} + (t + \bar{1})z) \cdot \bar{u} + y \ \& \ y < \bar{1} + (t + \bar{1})z).$$

Заметим, что свободными переменными формулы Φ_{Γ_0} являются x, z, t, y и имеет место эквивалентность:

$b = \Gamma_0(a, d, c)$ тогда и только тогда, когда в стандартной интерпретации истинна формула

$$(\Phi_{\Gamma_0})_{y,x,z,t}[\bar{b}, \bar{a}, \bar{d}, \bar{c}].$$

Поэтому функция $y = \Gamma_0(x, z, t)$ является арифметической.

Покажем, что оператор примитивной рекурсии, будучи примененным к арифметическим функциям, дает арифметическую функцию.

Пусть $n + 1$ -местная функция f получается из арифметических n -местной функции g и $n + 2$ -местной функции h применением оператора примитивной рекурсии, т.е. она удовлетворяет следующей системе равенств (схема примитивной рекурсии)

$$\begin{cases} f(x_1, \dots, x_n, 0) = g(x_1, \dots, x_n), \\ f(x_1, \dots, x_n, y + 1) = h(x_1, \dots, x_n, y, f(x_1, \dots, x_n, y)). \end{cases}$$

Предположим, что арифметические формулы Φ_g и Φ_h обеспечивают арифметичность функций $y = g(x_1, \dots, x_n)$ и $y = h(x_1, \dots, x_n, u, v)$. Тогда арифметичность функции $y = f(x_1, \dots, x_n, z)$ следует из эквивалентности

$$\begin{aligned} c = f(a_1, \dots, a_n, b) \iff & ((\bar{b} = \bar{0} \ \& \ (\Phi_g)_{y,x_1,\dots,x_n}[\bar{c}, \bar{a}_1, \dots, \bar{a}_n]) \vee \\ & (\bar{b} \neq \bar{0} \ \& \ (\exists x)(\exists z)(\exists y)((\Phi_g)_{y,x_1,\dots,x_n}[y, \bar{a}_1, \dots, \bar{a}_n] \ \& \ (\Phi_{\Gamma_0})_{y,x,z,t}[y, x, z, \bar{0}]) \ \& \\ & (\Phi_{\Gamma_0})_{y,x,z,t}[\bar{c}, x, z, \bar{b}]) \ \& \\ & (\forall t)(t < \bar{b} \rightarrow (\exists s)(\exists w)(\Phi_{\Gamma_0})_{y,x,z,t}[s, x, z, t] \ \& \ (\Phi_{\Gamma_0})_{y,x,z,t}[w, x, z, t + \bar{1}]) \ \& \\ & (\Phi_h)_{y,x_1,\dots,x_n,u,v}[w, \bar{a}_1, \dots, \bar{a}_n, s, t]) \end{aligned}$$

Случай, когда $n = 0$, т.е. схема примитивной рекурсии имеет вид

$$\begin{cases} f(0) = c, \\ f(y + 1) = h(y, f(y)), \end{cases}$$

где c — 0-местная функция, равная числу c , получается простым удалением в рассматриваемых формулах переменных x_1, \dots, x_n .

Остается доказать *арифметичность произвольного рекурсивно перечислимого множества U n -ок натуральных чисел.*

Как было установлено ранее, найдется примитивно рекурсивная функция $y = f(x_1, \dots, x_n, u)$ такая, что имеет место эквивалентность:

$(a_1, \dots, a_n) \in U$ тогда и только тогда,

когда $(\exists u)f(a_1, \dots, a_n, u) = 0$.

Значит $(a_1, \dots, a_n) \in U$ тогда и только тогда, когда формула

$$(\exists u)(\Phi_f)_{y, x_1, \dots, x_n, u}[\bar{0}, \bar{a}_1, \dots, \bar{a}_n, u]$$

истинна в стандартной интерпретации.

Поэтому U – арифметическое множество. \square

Рассмотрим вопрос о **нумерации арифметических формул**, т.е. формул языка $L_{\tau_{Arithm}}$ первого порядка сигнатуры τ_{Arithm} .

Напомним, алфавит $\Sigma_{\tau_{Arithm}}$ этого языка включает в себя счетное множество

$$V = \{x_1, x_2, \dots, x_n, \dots\}$$

индивидуальных переменных; четыре пропозициональные (логические) связки: отрицание \neg , конъюнкция $\&$, дизъюнкция \vee и импликация \rightarrow ; кванторы общности \forall и существования \exists , индивидуальные константы $\bar{0}$ и $\bar{1}$, которые мы ради краткости будем обозначать как 0 и 1 соответственно; двуместные функциональные символы $\bar{+}$ и $\bar{\cdot}$, которые мы ради краткости будем обозначать как $+$ и \cdot соответственно; двуместный предикатный символ $\bar{=}$, который обычно будем обозначать через $=$.

Занумеруем символы алфавита $\Sigma_{\tau_{Arithm}}$ нечетными натуральными числами, например, следующим образом с помощью функции γ , сопоставляющей каждому символу σ этого алфавита нечетное натуральное число $\gamma(\sigma)$

$$\begin{aligned} \gamma(()) &= 3, \gamma(()) = 5, \gamma(,) = 7, \\ \gamma(\neg) &= 9, \gamma(\&) = 11, \gamma(\vee) = 13, \gamma(\rightarrow) = 15, \\ \gamma(\forall) &= 17, \gamma(\exists) = 19, \\ \gamma(\bar{0}) &= 21, \gamma(\bar{1}) = 23, \\ \gamma(\bar{+}) &= 25, \gamma(\bar{\cdot}) = 27, \\ \gamma(\bar{=}) &= 29, \gamma(x_n) = 29 + 2n \ (n = 1, 2, \dots). \end{aligned}$$

Нумерацию символов алфавита, конечно, можно выполнить весьма различными способами. Требуется только, чтобы у разных символов были различные номера. Кроме того желательно, чтобы по номеру соответствующий символ "восстанавливался достаточно просто". Смысл этого пожелания будет ясен из дальнейшего.

Для нумерации слов в алфавите $\Sigma_{\tau_{Arithm}}$ предположим, что

$$p_0, p_1, \dots, p_n, \dots -$$

нумерация всех простых чисел в порядке возрастания. Таким образом $p_0 = 2$, $p_1 = 3$, $p_2 = 5$ и т.д.

Геделевым номером слова $\sigma_0\sigma_1\ldots\sigma_k$ в алфавите $\Sigma_{\tau_{Arithm}}$ называется число

$$g(\sigma_0\sigma_1\ldots\sigma_k) = p_0^{\gamma(\sigma_0)} p_1^{\gamma(\sigma_1)} \ldots p_k^{\gamma(\sigma_k)}.$$

По основной теореме арифметики каждое натуральное число, большее единицы, однозначно представимо в виде произведения степеней простых чисел. Поэтому номера различных слов различны. Кроме того, по произвольному натуральному числу легко определить, является ли оно номером некоторого слова и восстановить соответствующее слово (если оно существует). Геделевы номера слов – четные натуральные числа, поэтому они отличны от номеров символов алфавита, являющихся нечетными натуральными числами (впрочем, это не очень существенно). У символа σ алфавита два номера $\gamma(\sigma)$ и $g(\sigma)$.

Геделевым номером последовательности слов w_0, w_1, \ldots, w_q называется число

$$G(w_0, w_1, \ldots, w_q) = p_0^{g(w_0)} p_1^{g(w_1)} \ldots p_q^{g(w_q)}.$$

Геделевы номера последовательностей слов, будучи четными, отличны от геделевых номеров символов алфавита. Они отличны от геделевых номеров слов, так как последние содержат множитель 2 в нечетной степени, а у геделева номера последовательности слов показатель степени у 2 – четное натуральное число.

Конечно, номера различных последовательностей слов различны. Кроме того, по произвольному натуральному числу легко определить, является ли оно номером некоторой последовательности слов и восстановить соответствующую последовательность слов (если она существует).

У символа σ алфавита теперь три номера $\gamma(\sigma)$, $g(\sigma)$ и $G(\sigma)$, а у каждого слова w – два номера $g(w)$ и $G(w)$.

Замена символов алфавита, слов в этом алфавите и последовательностей слов ведет к **арифметизации языка** – замене выражений этого языка (термов, формул и т.д.) их номерами. Каждая формула языка $L_{\tau_{Arithm}}$ первого порядка сигнатуры τ_{Arithm} , как и любое слово, получает свой **геделев номер**. Обозначим через $\mathbf{Truth}_{\tau_{Arithm}}$ множество всех номеров замкнутых арифметических формул, т.е. формул языка $L_{\tau_{Arithm}}$, истинных в стандартной интерпретации. Нашей ближайшей целью будет доказательство знаменитой теоремы А. Тарского о "**неарифметичности истины**", утверждающей, что множество $\mathbf{Truth}_{\tau_{Arithm}}$ не является арифметическим.

Теорема 1.2 (А. Тарский). Множество $\mathbf{Truth}_{\tau_{Arithm}}$ всех номеров замкнутых арифметических формул, т.е. формул языка $L_{\tau_{Arithm}}$, истинных в стандартной интерпретации, не является арифметическим.

Предварительно докажем ряд вспомогательных утверждений, позволяющих преобразования формул заменить преобразованиями их номеров. Прежде всего установим примитивную рекурсивность или рекурсивность ряда необходимых

для дальнейшего отношений (предикатов) и функций. Одноместные отношения будем называть свойствами. Напомним, что классы примитивно рекурсивных и рекурсивных отношений и предикатов замкнуты относительно логических операций и навешивания ограниченных кванторов.

(1) $WIC(x)$ – " x – геделев номер слова, являющегося индивидуальной константой". Это свойство задается формулой

$$(x = 2^{21} \vee x = 2^{23}).$$

Поэтому оно примитивно рекурсивно.

(2) $WFL(x)$ – " x – геделев номер слова, являющегося функциональным символом". Это свойство задается формулой

$$(x = 2^{25} \vee x = 2^{27}).$$

Поэтому оно примитивно рекурсивно.

(3) $WPL(x)$ – " x – геделев номер слова, являющегося предикатным символом". Это свойство задается формулой

$$x = 2^{29}.$$

Поэтому оно примитивно рекурсивно.

(4) $WVbl(x)$ – " x – геделев номер слова, являющегося индивидуальной переменной". Это свойство задается формулой

$$(\exists u)_{u < x} (1 \leq u \ \& \ x = 2^{29+2u}).$$

Поэтому оно примитивно рекурсивно.

Учитывая специфику языка $L_{\tau_{Arithm}}$ первого порядка сигнатуры τ_{Arithm} (все входящие в эту сигнатуру функциональные и предикатные символы являются 2-местными), несколько изменим определение понятий терма и формулы этого языка.

Определение 1.4. 1. Каждая индивидуальная переменная x_n является термом.
2. Каждая индивидуальная константа $\bar{0}$ и $\bar{1}$ является термом.
3. Если t_1 и t_2 – термы, то следующие слова являются термами

$$(t_1 \bar{+} t_2), \quad (t_1 \bar{\cdot} t_2).$$

Определение 1.5. 1. Если t_1 и t_2 – термы, то следующее слово является формулой

$$(t_1 \bar{=} t_2).$$

Формулы указанного вида называются элементарными или базовыми формулами.

2. Если \mathcal{A} и \mathcal{B} – формулы, то следующие слова являются формулами

$$(\neg \mathcal{A}), \quad (\mathcal{A} \ \& \ \mathcal{B}), \quad (\mathcal{A} \vee \mathcal{B}), \quad (\mathcal{A} \rightarrow \mathcal{B}),$$

$$(\forall x_n) \mathcal{A}, \quad (\exists x_n) \mathcal{A},$$

где x_n – произвольная индивидуальная переменная.

Нашей ближайшей целью будет установление примитивной рекурсивности следующих двух свойств:

$Trm(x)$ – " x – геделев номер слова, являющегося термом" и

$Fml(x)$ – " x – геделев номер слова, являющегося формулой".

Ранее была установлена примитивная рекурсивность функций $ex(x, y)$ – экспонента простого числа p_x в числе y , равная наибольшему показателю степени, в которой простое число p_x делит число y ; $lh(x)$ – число ненулевых показателей в каноническом разложении числа x на простые множители (по определению $lh(0) = 0$); $long(x)$ – наибольшее натуральное число t такое, что простое число p_t делит x (по определению $long(0) = 0$) и функция $x * y$, которая по номерам двух последовательностей положительных натуральных чисел вычисляет номер их конкатенации, т.е. если $a = (a_0, a_1, \dots, a_k)$ и $b = (b_0, b_1, \dots, b_m)$ – последовательности натуральных чисел, $n(a) = p_0^{a_0} p_1^{a_1} \dots p_k^{a_k}$, $n(b) = p_0^{b_0} p_1^{b_1} \dots p_m^{b_m}$ и

$$a \cdot b = (a_0, a_1, \dots, a_k, b_0, b_1, \dots, b_m),$$

то

$$n(a) * n(b) = n(a \cdot b) = p_0^{a_0} p_1^{a_1} \dots p_k^{a_k} \cdot p_{k+1}^{b_0} p_{k+2}^{b_1} \dots p_{k+m+1}^{b_m}.$$

Другими словами, для любых двух слов W и U для их геделевых номеров выполняется равенство $g(WU) = g(W) * g(U)$, где WU – произведение (соединение, конкатенация) слов W и U .

Кроме того, нам потребуется следующий установленный выше факт.

По произвольной $n + 1$ -местной функции f была определена функция

$$f^*(x_1, \dots, x_n, y) = \prod_{i=0}^y p_i^{f(x_1, \dots, x_n, i)}.$$

и установлено, что функция f примитивно рекурсивна, рекурсивна или частично рекурсивна тогда и только тогда, когда примитивно рекурсивна, рекурсивна или частично рекурсивна функция f^* .

Рассмотрим близкую к функции f^* функцию $f_{\#}$, заданную равенствами

$$f_{\#}(x_1, \dots, x_n, 0) = 1, \quad f_{\#}(x_1, \dots, x_n, y + 1) = \prod_{i=0}^y p_i^{f(x_1, \dots, x_n, i)}.$$

Легко понять, что функция f примитивно рекурсивна, рекурсивна или частично рекурсивна тогда и только тогда, когда примитивно рекурсивна, рекурсивна или частично рекурсивна функция $f_{\#}$.

Теорема 1.3. Если $n + 2$ -местная функция $h(x_1, \dots, x_n, y, z)$ примитивно рекурсивна, рекурсивна или частично рекурсивна, а $n + 1$ -местная функция $f(x_1, \dots, x_n, y)$ удовлетворяет равенству

$$f(x_1, \dots, x_n, y) = h(x_1, \dots, x_n, y, f_{\#}(x_1, \dots, x_n, y)),$$

то функция $f(x_1, \dots, x_n, y)$ соответственно примитивно рекурсивна, рекурсивна или частично рекурсивна.

Д о к а з а т е л ь с т в о. Из равенств

$$\begin{aligned} f_{\#}(x_1, \dots, x_n, 0) &= 1, \\ f_{\#}(x_1, \dots, x_n, y+1) &= f_{\#}(x_1, \dots, x_n, y) \cdot p_y^{f(x_1, \dots, x_n, y)} = \\ &= f_{\#}(x_1, \dots, x_n, y) \cdot p_y^{h(x_1, \dots, x_n, y, f_{\#}(x_1, \dots, x_n, y))} \end{aligned}$$

следует:

функция $f_{\#}(x_1, \dots, x_n, y)$ примитивно рекурсивна, рекурсивна или частично рекурсивна в зависимости от того, каковы функции

$$h(x_1, \dots, x_n, y, z) \text{ и } f(x_1, \dots, x_n, y).$$

Для доказательства примитивной рекурсивности, рекурсивности или частичной рекурсивности функции $f(x_1, \dots, x_n, y)$ при условии, что таковы функции $h(x_1, \dots, x_n, y, z)$ и $f_{\#}(x_1, \dots, x_n, y)$, достаточно воспользоваться равенством

$$f(x_1, \dots, x_n, y) = ex(y, f_{\#}(x_1, \dots, x_n, y+1)).$$

□

По произвольному отношению $S(x_1, \dots, x_n, y, z)$ определим новое отношение $R(x_1, \dots, x_n, y)$ с помощью эквивалентности

$$R(x_1, \dots, x_n, y) \iff S(x_1, \dots, x_n, y, (\chi_R)_{\#}(x_1, \dots, x_n, y)).$$

Следствие 1. *Если отношение $S(x_1, \dots, x_n, y, z)$ примитивно рекурсивно, рекурсивно или частично рекурсивно, то таково же и отношение $R(x_1, \dots, x_n, y)$.*

Д о к а з а т е л ь с т в о сразу следует из предыдущей теоремы, если заметить, что характеристическая функция $(\chi_R)(x_1, \dots, x_n, y)$ отношения $R(x_1, \dots, x_n, y)$ удовлетворяет равенству

$$(\chi_R)(x_1, \dots, x_n, y) = (\chi_S)(x_1, \dots, x_n, y, (\chi_R)_{\#}(x_1, \dots, x_n, y)).$$

□

Заметим, что выполнимость $R(x_1, \dots, x_n, y)$ эквивалентна равенству $\chi_R(x_1, \dots, x_n, y) = 1$. Последнее равенство при любом $z > y$ эквивалентно равенству $ex(y, (\chi_R)_{\#}(x_1, \dots, x_n, z)) = 1$.

(5) Для установления примитивной рекурсивности свойства $Trm(x)$ — " x — геделев номер слова, являющегося термом" воспользуемся тем, что оно эквивалентно свойству

$$\begin{aligned} WIC(x) \vee WVbl(x) \vee \\ (\exists u)_{u < x} (\exists v)_{v < x} (\exists t)_{t < x} (Trm(u) \& Trm(v) \& WFL(t) \& \\ x = 2^3 * u * t * v * 2^5). \end{aligned}$$

Рассмотрим примитивно рекурсивное отношение $S(x, z)$, заданное формулой

$$WIC(x) \vee WVbl(x) \vee \\ (\exists u)_{u < x} (\exists v)_{v < x} (\exists t)_{t < x} (ex(u, z) = 1 \& ex(v, z) = 1 \& WFL(t) \& \\ x = 2^3 * u * t * v * 2^5).$$

Тогда свойство $Trm(x)$ равносильно $S(x, (\chi_{Trm})_{\#}(x))$

(6) Для установления примитивной рекурсивности свойства $Atmfl(x)$ – “ x – геделев номер атомарной арифметической формулы” воспользуемся тем, что атомарные арифметические формулы имеют вид $(t_1 \doteq t_2)$, где t_1 и t_2 – произвольные термы. Поэтому свойство $Atmfl(x)$ эквивалентно свойству

$$(\exists u)_{u < x} (\exists v)_{v < x} (Trm(u) \& Trm(v) \& x = 2^3 * u * 2^{29} * v * 2^5).$$

(7) Для установления примитивной рекурсивности свойства $Fmfl(x)$ – “ x – геделев номер арифметической формулы” воспользуемся тем, что оно эквивалентно свойству

$$(Atmfl(x) \vee \exists u)_{u < x} (Fmfl(u) \& x = 2^3 * 2^9 * u * 2^5) \vee \\ (\exists u)_{u < x} (\exists v)_{v < x} (Frml(u) \& Frml(v) \& \\ (x = 2^3 * u * 2^{11} * v * 2^5 \vee x = 2^3 * u * 2^{13} * v * 2^5 \vee \\ x = 2^3 * u * 2^{15} * v * 2^5)) \vee \\ (\exists u)_{u < x} (\exists v)_{v < x} (WVbl(u) \& Frml(v) \& \\ (x = 2^3 * 2^{17} * u * 2^5 * v \vee (x = 2^3 * 2^{19} * u * 2^5 * v))).$$

(8) В этом пункте будет установлена примитивная рекурсивность нескольких предикатов, связанных с подстановкой термов в термы и формулы вместо свободных вхождений переменных. Мы будем устанавливать это в соответствии с индуктивными определениями термов и формул. Конечно, некоторые формулы можно было бы сократить, но при этом, на наш взгляд, несколько усложнилось бы понимание их “смысла” для начинающего изучать эту технику, используемую в доказательстве теорем Г. Геделя и А. Тарского. Выбирая между краткостью изложения и легкостью понимания мы сознательно предпочли последнее, облегчив тем самым, как нам кажется, понимание смысла формальных конструкций для начинающего изучать математическую логику.

(8.1) Примитивно рекурсивен предикат $Subst_{Fml}^{(1)}(\gamma, u, v)$ – “ $ex(0, \gamma)$ – геделев номер результата подстановки в формулу с геделевым номером $ex(1, \gamma)$ терма с геделевым номером u вместо всех свободных вхождений переменной с геделевым номером v ”.

Чтобы несколько уменьшить размеры рассматриваемых формул предварительно установим примитивную рекурсивность двух вспомогательных предикатов $Subst_{Trm}(\gamma, u, v)$ – “ $ex(1, \gamma)$ – геделев номер терма, а $ex(0, \gamma)$ – геделев

номер результата подстановки в этот терм терма с геделевым номером u вместо всех вхождений переменной с геделевым номером v .”

$$\begin{aligned}
 & Trm(u) \& WVbl(v) \& Trm(ex(1, \gamma)) \& \\
 & ((\exists z)_{z < ex(1, \gamma)} (ex(1, \gamma) = 2^z \& \\
 & (ex(1, \gamma) = v \& ex(0, \gamma) = u) \vee (ex(1, \gamma) \neq v \& ex(0, \gamma) = ex(1, \gamma))) \vee \\
 & (\exists \alpha)_{\alpha < ex(1, \gamma)} (\exists \beta)_{\beta < ex(1, \gamma)} (\exists \delta)_{\delta < ex(1, \gamma)} (WFL(\delta) \& Trm(\alpha) \& Trm(\beta) \& \\
 & ex(1, \gamma) = 2^3 * \alpha * \delta * \beta * 2^5 \& \\
 & (\exists \alpha_0)_{\alpha_0 < ex(0, \gamma)} (\exists \beta_0)_{\beta_0 < ex(0, \gamma)} (ex(0, \gamma) = 2^3 * \alpha_0 * \delta * \beta_0 * 2^5 \& \\
 & Subst_{Trm}(2^{\alpha_0} \cdot 3^\alpha, u, v) \& Subst_{Trm}(2^{\beta_0} \cdot 3^\beta, u, v))) .
 \end{aligned}$$

$Subst_{Atmfl}(\gamma, u, v)$ – ” $ex(1, \gamma)$ – геделев номер элементарной формулы, а $ex(0, \gamma)$ – геделев номер результата подстановки в эту формулу терма с геделевым номером u вместо всех вхождений переменной с геделевым номером v .”

$$\begin{aligned}
 & Trm(u) \& WVbl(v) \& Atmfl(ex(1, \gamma)) \& \\
 & (\exists \alpha)_{\alpha < ex(1, \gamma)} (\exists \beta)_{\beta < ex(1, \gamma)} (Trm(\alpha) \& Trm(\beta) \& \\
 & ex(1, \gamma) = 2^3 * \alpha * 2^{29} * \beta * 2^5 \& \\
 & (\exists \alpha_0)_{\alpha_0 < ex(0, \gamma)} (\exists \beta_0)_{\beta_0 < ex(0, \gamma)} (ex(0, \gamma) = 2^3 * \alpha_0 * 2^{29} * \beta_0 * 2^5 \& \\
 & Subst_{Trm}(2^{\alpha_0} \cdot 3^\alpha, u, v) \& Subst_{Trm}(2^{\beta_0} \cdot 3^\beta, u, v)))
 \end{aligned}$$

$Subst_{Fml}^{(1)}(\gamma, u, v)$ – ” $ex(1, \gamma)$ – геделев номер формулы, а $ex(0, \gamma)$ – геделев номер результата подстановки в эту формулу терма с геделевым номером u вместо всех свободных вхождений переменной с геделевым номером v .”

$$\begin{aligned}
 & Fml(ex(1, \gamma)) \& (Atmfl(ex(1, \gamma)) \& Subst_{Atmfl}(\gamma, u, v)) \vee \\
 & ((\exists \alpha)_{\alpha < ex(1, \gamma)} (\exists \alpha_0)_{\alpha_0 < ex(0, \gamma)} (Fml(\alpha) \& \\
 & ex(1, \gamma) = 2^3 * 2^9 * \alpha * 2^5 \& ex(0, \gamma) = 2^3 * 2^9 * \alpha_0 * 2^5 \& \\
 & Subst_{Fml}(2^{\alpha_0} \cdot 3^\alpha, u, v)) \vee \\
 & ((\exists \alpha)_{\alpha < ex(1, \gamma)} (\exists \alpha_0)_{\alpha_0 < ex(0, \gamma)} (\exists \beta)_{\beta < ex(1, \gamma)} (\exists \beta_0)_{\beta_0 < ex(0, \gamma)} (\exists \delta)_{\delta < ex(1, \gamma)} \\
 & ((\delta = 2^{11} \vee \delta = 2^{13} \vee \delta = 2^{15}) \& (Fml(\alpha) \& Fml(\beta) \& \\
 & ex(1, \gamma) = 2^3 * \alpha * \delta * \beta * 2^5 \& ex(0, \gamma) = 2^3 * \alpha_0 * \delta * \beta_0 * 2^5 \& \\
 & Subst_{Fml}(2^{\alpha_0} \cdot 3^\alpha, u, v) \& Subst_{Fml}(2^{\beta_0} \cdot 3^\beta, u, v)) \vee \\
 & ((\exists \alpha)_{\alpha < ex(1, \gamma)} (\exists \alpha_0)_{\alpha_0 < ex(0, \gamma)} (\exists \beta)_{\beta < ex(1, \gamma)} (\exists \beta_0)_{\beta_0 < ex(0, \gamma)} (\exists \delta)_{\delta < ex(1, \gamma)} \\
 & ((\delta = 2^{17} \vee \delta = 2^{19}) \& (Fml(\alpha) \& WVbl(\beta) \& \\
 & ex(1, \gamma) = 2^3 * \delta * \beta * 2^5 * \alpha \& ex(0, \gamma) = 2^3 * \delta * \beta * \alpha_0 * 2^5 \& \\
 & ((\beta = v \& \alpha_0 = \alpha) \vee (\beta \neq v \& Subst_{Fml}(2^{\alpha_0} \cdot 3^\alpha, u, v))) .
 \end{aligned}$$

(8.2) Примитивно рекурсивен предикат $Subst_{Fml}(x, y, u, v)$ – ” x – геделев номер результата подстановки в формулу с геделевым номером y терма с геделевым номером u вместо всех свободных вхождений переменной с геделевым номером v .”

Ясно, что $Subst_{Fml}(x, y, u, v)$ эквивалентно $Subst_{Fml}^{(1)}(2^x \cdot 3^y, u, v)$

(8.3) Примитивная рекурсивность функции $Sub(y, u, v)$ – ”геделев номер результата подстановки в выражение с геделевым номером y терма с геделевым номером u вместо всех свободных вхождений переменной с геделевым номером v ” следует из равенства

$$Sub(y, u, v) = \mu x_{x < p_{uy}^{uy} Subst(x, y, u, v)}.$$

(9) Установим примитивную рекурсивность предиката $Fr(u, v)$ ” u – геделев номер формулы, содержащей свободные вхождения переменной с геделевым номером v и не содержащий свободных вхождений других переменных”.

Это следует из его эквивалентности предикату

$$(Fml(u) \& WVbl(v) \& (\neg Subst_{Fml}(u, u, 2^{21}, v) \& (\forall x)_{x < u} (x \neq v \rightarrow Subst_{Fml}(u, u, 2^{21}, x))))$$

Заметим, что если переменная с геделевым номером x входит в слово с номером u , то $x < u$.

Содержательный смысл последней формулы

”если в формулу с геделевым номером u вместо всех свободных вхождений переменной с геделевым номером v подставить константу $\bar{0}$, то полученная формула будет отлична от исходной, а если в эту формулу вместо всех свободных вхождений переменной с геделевым номером x , отличным от v , подставить константу $\bar{0}$, то полученная формула совпадет с исходной”.

(10) Примитивно рекурсивна функция φ , которая по натуральному числу n дает геделев номер терма \bar{n} , т.е. $\varphi(n) = g(\bar{n})$. Это следует из равенств

$$\begin{aligned} \varphi(0) &= 2^{21}, \\ \varphi(n + 1) &= 2^3 * \varphi(n) * 2^{25} * 2^{23} * 2^5. \end{aligned}$$

Д о к а з а т е л ь с т в о теоремы А. Тарского проведем методом от противного. Предположим, что множество всех номеров замкнутых арифметических формул, т.е. формул языка $L_{\tau_{Arithm}}$, истинных в стандартной интерпретации, является арифметическим. Значит *существует формула **Truth** с одной свободной переменной x_1 такая, что для произвольного натурального числа n справедлива эквивалентность:*

*формула **Truth** $_{x_1}[\bar{n}]$ истинна в стандартной интерпретации тогда и только тогда, когда натуральное число n является геделевым номером замкнутой арифметической формулы, истинной в стандартной интерпретации.*

Построим формулу P с двумя свободными переменными x_1 и x_2 такую, что для произвольных натуральных чисел n и m справедлива эквивалентность:

формула $P_{x_1, x_2}[\bar{n}, \bar{m}]$ истинна в стандартной интерпретации тогда и только тогда, когда n – геделев номер некоторой арифметической формулы F с одной свободной переменной x_1 и формула $F_{x_1}[\bar{m}]$ ложна в стандартной интерпретации.

В качестве искомой формулы P можно взять следующую

$$(Fr(x_1, 2^{31}) \& (\exists x_3)(\exists x_4)(x_4 = \varphi(x_2) \& Subst_{Fml}(x_3, x_1, x_4, 2^{31}) \& (\neg \mathbf{Truth}_{x_1}[x_3])))).$$

Для завершения доказательства теоремы А. Тарского обозначим через p *геделев номер формулы* $P_{x_1, x_2}[x_1, x_1]$ с одной свободной переменной x_1 . Эту формулу ради некоторой краткости будем обозначать через D . Тогда

формула $D_{x_1}[\bar{p}] = D_{x_1, x_2}[\bar{p}, \bar{p}]$ истинна в стандартной интерпретации тогда и только тогда, когда формула $D_{x_1}[\bar{p}]$ ложна в стандартной интерпретации.

Полученное противоречие завершает доказательство теоремы А. Тарского.

□

Следствие 2. *Множество $\mathbf{Truth}_{\tau_{Arithm}}$ всех номеров замкнутых арифметических формул, истинных в стандартной интерпретации, не является рекурсивно перечислимым, а значит, оно и не рекурсивно.*

Следствие 3. *Невозможен алгоритм, позволяющий по произвольной замкнутой арифметической формуле определить, истинна ли она в стандартной интерпретации.*

§2. Проблема выводимости для полусистем Туэ и комбинаторная проблема Поста

В этом разделе рассматриваются некоторые алгоритмические проблемы, связанные с полусистемами и системами Туэ, системами Поста и формальными грамматиками. Фундаментальную роль при этом будет играть установленная выше *алгоритмическая неразрешимость проблемы останова* для машин Тьюринга – для доказательства неразрешимости некоторой алгоритмической проблемы мы будем сводить к ней проблему останова.

По *машине Тьюринга* M с *внешним алфавитом*, или *алфавитом ленточных символов*,

$$\mathcal{A}_M = \{a_0, a_1, \dots, a_n\},$$

внутренним алфавитом, или *алфавитом внутренних состояний*,

$$\mathcal{Q}_M = \{q_0, q_1, \dots, q_m\}$$

и *программой* P_M естественным образом построим полусистему Туэ $SST(M)$ над алфавитом

$$A = \{a_0, a_1, \dots, a_n, q_0, q_1, \dots, q_m, h\}$$

с множеством продукций Π , которое следующим образом строится по программе P_M .

Для каждой команды $P_M(i, j)$ из P_M мы включаем в Π продукцию определенного ниже вида.

I. Если $P_T(i, j)$ имеет вид $q_i a_j \rightarrow q_r a_t S$, то включаем в Π одну продукцию $q_i a_j \rightarrow q_r a_t$.

II. Если команда $P_T(i, j)$ имеет вид $q_i a_j \rightarrow q_r a_t L$, то включаем все продукции вида $a_s q_i a_j \rightarrow q_r a_s a_t$ при $s = 0, \dots, n$ и еще одну продукцию $h q_i a_j \rightarrow h q_r a_0 a_t$.

III. Если команда $P_T(i, j)$ имеет вид $q_i a_j \rightarrow q_r a_t R$, то включаем все продукции вида $q_i a_j a_s \rightarrow a_t q_r a_s$ при $s = 0, \dots, n$ и еще одну продукцию $q_i a_j h \rightarrow a_t q_r a_0 h$.

IV. Кроме того, добавим в Π еще продукцию $h q_0 h \rightarrow q_0$ и все продукции вида $q_0 a_t \rightarrow q_0$, $a_t q_0 \rightarrow q_0$.

Каждой конфигурации $X = E_1 q_i E_2$ машины T ставим в соответствие слово $h E_1 q_i E_2 h$, которое будем называть *словом Поста*.

Заметим, что элементарные преобразования, соответствующие продукциям типов I, II, III, любое слово Поста переводят в слово Поста. Следующая лемма очевидна.

Лемма 2.1. *Машина Тьюринга T переводит за один такт работы конфигурацию X в конфигурацию Y в том и только том случае, если слово Поста hXh переводится в слово Поста hYh одним элементарным преобразованием, соответствующим команде, применяемой машиной T .*

Лемма 2.2. *Машина Тьюринга T , начав работать в конфигурации X , через конечное число тактов работы остановится тогда и только тогда, когда в полусистеме $\text{Туэ } SST(M)$ из слова hXh выводимо слово q_0 , т.е.*

$$hXh \vdash_{SST(M)}^* q_0.$$

До к а з а т е л ь с т в о. Пусть машина Тьюринга T , начав работать в конфигурации X , через конечное число тактов работы остановится, т.е. имеется конечная цепочка последовательных конфигураций

$$X, X_1, X_2, \dots, X_{k-1}, X_k,$$

оканчивающаяся заключительной конфигурацией X_k . Тогда по лемме 2.1 имеем последовательность элементарных преобразований полусистемы $\text{Туэ } ST(M)$

$$hXh \rightarrow hX_1h \rightarrow hX_2h \rightarrow \dots \rightarrow hX_{k-1}h \rightarrow hX_kh,$$

где hX_kh имеет вид $hZ_1q_0Z_2h$. Используя продукции из IV-ой группы, получаем, что в полусистеме $\text{Туэ } SST(M)$

$$hXh \vdash_{SST(M)}^* q_0.$$

Предположим теперь, что в полусистеме $\text{Туэ } SST(M)$

$$hXh \vdash_{SST(M)}^* q_0.$$

Это значит, что существует цепочка элементарных преобразований

$$hXh \equiv Y_0 \rightarrow Y_1 \rightarrow Y_2 \rightarrow \dots \rightarrow Y_{k-1} \rightarrow Y_k \equiv q_0. \quad (1)$$

Очевидно, в каждое слово Y_j из последовательности (1) входит только одна q -буква. Рассмотрим кратчайшую цепочку элементарных преобразований вида (1), заканчивающуюся словом, содержащим q_0 :

$$hXh \equiv Y_0 \rightarrow Y_1 \rightarrow Y_2 \rightarrow \dots \rightarrow Y_{r-1} \rightarrow Y_r \equiv Z_1 q_0 Z_2. \quad (2)$$

В преобразованиях (2) продукции полусистемы Туэ $SST(M)$ из IV-ой группы не используются. Поэтому в (2) все слова являются словами Поста.

Тогда по лемме 2.1 машина T из конфигурации X через конечное число шагов перейдет в заключительное состояние q_0 , т.е. остановится. Лемма 2.2 доказана. \square

Пусть T_0 – машина Тьюринга с неразрешимой проблемой остановки. Тогда из лемм 2.1 и 2.2 следует

Теорема 2.1. *Невозможен алгоритм, определяющий для произвольного слова X в алфавите полусистемы Туэ $SST(M)$, верно ли*

$$hXh \vdash_{SST(M)}^* q_0.$$

В заключение параграфа рассмотрим некоторые алгоритмические вопросы, связанные с системами Поста и формальными грамматиками.

Со времен Древней Греции основным методом изложения математических теорий служит *аксиоматический метод*, знакомый каждому, изучавшему в свое время школьный курс геометрии Евклида. Коротко суть аксиоматического метода можно выразить следующим образом: выделяются *основные, неопределяемые понятия*, в геометрии – это “точка”, “прямая” и “плоскость”, вводятся некоторые неопределяемые отношения, в геометрии – это, например, “точка A принадлежит прямой L ”, “точка A принадлежит плоскости L ”, “точка C находится на прямой L между точками B и C ” и (или) ряд других, в зависимости от реализуемой концепции. Затем формулируется ряд утверждений относительно этих понятий и отношений, которые называются “*аксиомами*”, в геометрии – это, например, утверждение о том, что для любых двух точек существует единственная прямая, которой принадлежит каждая из этих точек, обычно говорят, что прямая проходит через эти две точки; из любых трех различных точек прямой одна находится между двумя другими; для любых трех различных точек, не лежащих на одной прямой, существует единственная плоскость, которой принадлежит каждая из этих трех точек; аксиома параллельности и т.д. Нередко аксиома о существовании единственного объекта с некоторыми свойствами разбивается на две аксиомы, в одной из которых утверждается, что объект существует, а во второй – что существует не более одного такого объекта. Яркие примеры исчерпывающих аксиоматизаций изложены в знаменитой

монографии Д. Гильберта "Основания геометрии". Затем из аксиом путем (с помощью) *логических рассуждений* выводятся (доказываются) теоремы. При этом на протяжении более чем двух тысяч лет не возникало необходимости точно определить смысл слов *логическое рассуждение* – со времен Евклида и до конца XIX века основное внимание уделялось формированию списка аксиом для тех или иных теорий, прежде всего так или иначе связанных с геометрией. Правила доказательства теорем, их вывода из аксиом были неформальными, основанными на "здравом смысле". Кроме того, долгое время на аксиомы смотрели как на точные формулировки "интуитивно очевидных утверждений". Лишь после открытия (создания) в XIX веке неевклидовых геометрий, точнее после постепенного осознания "права на существование" в качестве аксиом "несколько необычных для нас утверждений" о, казалось бы, "хорошо нам известных понятиях", поведение которых нам "интуитивно ясно", хотя и не могло быть выведено из уже сформулированных аксиом, стал выкристаллизовываться вопрос о том, что же такое аксиома и еще более сложный и важный вопрос, что такое *логическое рассуждение*. С начала XX века начались интенсивные исследования в области оснований математики. В значительной степени это стимулировалось открытием парадоксов (противоречий) в ряде разделов математики и неформальной логики. Было осознано, что "здравый смысл" не может служить надежным средством исследования достаточно тонких вопросов, относящихся к основаниям математики. Возникла необходимость в тщательном исследовании самих понятий *математическое доказательство* и *логическое рассуждение* с целью формализации всех этапов дедуктивного вывода. В этих исследованиях все большую роль играло понятие *эффективной процедуры*. Постепенно это привело к понятию *логической системы*.

Логистическая или формальная система L включает в качестве своих основных частей *множество аксиом* и *множество правил вывода*.

Аксиома – это просто слово специального вида в алфавите системы L . Особую роль играют системы с *эффективным понятием аксиомы*, т.е. в этом случае *предполагается наличие алгоритма, позволяющего по произвольному слову в алфавите системы ответить на вопрос, является ли оно аксиомой*.

Правило вывода – это $n + 1$ -местное отношение (предикат)

$$R(w_1, \dots, w_n, w_{n+1})$$

на множестве всех слов в алфавите системы. При этом если,

$$R(w_1, \dots, w_n, w) = \text{И},$$

то говорят, что

слово w получается (следует) из слов w_1, \dots, w_n по правилу R , и пишут

$$w_1, \dots, w_n \xrightarrow{R} w$$

или

$$\frac{w_1, \dots, w_n}{w} R.$$

При этом слова w_1, \dots, w_n называются *посылками* правила R , слово w – его *заключением*.

И вновь обычно *предполагается наличие алгоритма, позволяющего по произвольным словам w_1, \dots, w_n и w в алфавите системы ответить на вопрос, получается ли слово w из слов w_1, \dots, w_n по правилу R* . Мы будем рассматривать лишь системы с *конечными множествами правил вывода*.

Выводом или **доказательством** в системе L называется любая конечная последовательность слов u_1, \dots, u_m в алфавите этой системы такая, что для любого i

либо u_i – аксиома системы L ,

либо слово u_i получается из некоторых предшествующих ему в этой последовательности слов u_{j_1}, \dots, u_{j_n} по некоторому $n + 1$ -местному правилу R т.е. $j_1, \dots, j_n < i$.

Слово u называется **выводимым** в системе L или **теоремой** этой системы, если существует вывод (доказательство) в системе L , оканчивающийся (оканчивающееся) словом u .

Запись $\vdash_L u$ будет служить сокращением для утверждения "слово u выводимо в системе L ".

Легко понять, что *существует алгоритм, позволяющий по любой конечной последовательности слов системы L с эффективными понятиями аксиомы и правила вывода ответить на вопрос, является ли эта последовательность выводом (доказательством)*.

Весьма важными примерами логистических систем являются исчисление высказываний и исчисление предикатов конечной сигнатуры.

Для логистической системы L с эффективными понятиями аксиомы и правила вывода естественно возникает *вопрос об эффективности понятия выводимости (доказуемости) слова в данной логистической системе L* , т.е. вопрос о существовании алгоритма, позволяющего по произвольному слову w в алфавите системы L ответить на вопрос, выводимо ли слово w в системе L . Легко построить алгоритм, эффективно генерирующий все доказательства. Поэтому если слово w выводимо в системе L , то мы это установим, а если нет, то наша процедура не даст ответа. Для логистической системы L с эффективными понятиями аксиомы и правила вывода множество выводимых слов является рекурсивно перечислимым, однако может не быть рекурсивным.

Если для логистической системы L существует алгоритм, позволяющий по произвольному слову w в алфавите системы L ответить на вопрос, выводимо ли слово w в системе L , то говорят, что L – логистическая система с эффективным понятием теоремы. Важными примерами таких систем служат исчисление высказываний, исчисление одноместных предикатов, элементарная евклидова геометрия, арифметика Пресбургера и ряд других.

Для произвольного конечного множества слов $\{w_1, \dots, w_n\}$ в алфавите логистической системы L обозначим через $L(w_1, \dots, w_n)$ систему, полученную из L добавлением слов w_1, \dots, w_n к ее аксиомам.

Слово u называется *выводимым в системе L из множества слов*

$\{w_1, \dots, w_n\}$, если слово u выводимо в системе $L(w_1, \dots, w_n)$.

Запись $w_1, \dots, w_n \vdash_L u$ будет служить сокращением для утверждения ”слово u выводимо в системе L из множества слов $\{w_1, \dots, w_n\}$ ”.

Глубокий анализ вычислительной деятельности в широком смысле этого слова, проделанный А. Тьюрингом, Э. Постом и рядом других исследователей, привел к весьма интересным и достаточно неожиданным выводам – *значительная часть математической деятельности предполагает весьма ограниченный набор способностей у исследователя:*

1) способность просматривать побуквенно слово, выделять в нем некоторые фиксированные подслова, фиксированные подпоследовательности идущих друг за другом букв,

2) способность извлекать из слова подслова фиксированного вида, сохраняя оставшиеся части слова,

3) способность перестраивать части слова – заменять подслова на другие слова, вставлять подслова фиксированного вида, удалять подслова.

Одним из основных понятий для нас будет понятие **продукции**, которое мы понимаем как *точное правило, указывающее, как разбивать слово на части и перестраивать эти части.*

Под **продукцией Поста** мы будем понимать любое правило преобразования слов, задаваемое схемой вида

$$u_0 W_1 u_1 W_2 \dots W_n u_n \rightarrow v_0 V_1 v_1 V_2 \dots V_m v_m,$$

где $u_0, u_1, \dots, u_n, v_0, v_1, \dots, v_m$ – фиксированные для данной продукции слова, причем u_0 или u_n может быть пустым словом, некоторые из слов v_0, v_1, \dots, v_m могут быть пустыми;

W_1, \dots, W_n – произвольные слова, их часто называют переменными словами,

V_1, \dots, V_m – фиксированные слова в алфавите $\{W_1, \dots, W_n\}$.

Системой Поста называется логистическая система L в некотором алфавите \mathcal{A} , задаваемая набором аксиом, являющихся словами в алфавите \mathcal{A} , и набором продукций, постоянные слова которых – слова в алфавите \mathcal{A} .

Легко понять, что полусистемы и системы Туэ (подстановочные системы) – частный случай систем Поста.

Можно показать, что системы Поста дают достаточно адекватный аппарат для описания работы машин Тьюринга.

Мы не будем заниматься подробным изучением этого вопроса, отдав предпочтение полусистемам и системам Туэ. Рассмотрим лишь **канонические системы Поста** – системы Поста с одной аксиомой и *нормальными произведениями* – произведениями вида

$$uW \rightarrow Wv.$$

Установим некоторую связь между полусистемами Туэ и каноническими системами Поста.

Для произвольной полусистемы Туэ SST

$$SST = \langle a_1, \dots, a_n \mid A_1 \rightarrow B_1, \dots, A_m \rightarrow B_m \rangle$$

обозначим через $CSP(SST)$ следующую каноническую систему Поста

$$CSP(SST) = \langle a_1, \dots, a_n, a'_1, \dots, a'_n \mid \\ A_1 W \rightarrow W B'_1, \dots, A_m W \rightarrow W B'_m, \\ a_1 W \rightarrow W a'_1, \dots, a_n W \rightarrow W a'_n, \\ a'_1 W \rightarrow W a_1, \dots, a'_n W \rightarrow W a_n \rangle,$$

где для произвольного слова $U = a_{i_1} \dots a_{i_p}$ через U' обозначено слово $a'_{i_1} \dots a'_{i_p}$.

Справедлива следующая теорема.

Теорема 2.2. *Для любых двух слов U и V в алфавите полусистемы Туэ SST справедлива эквивалентность*

$$U \vdash_{SST} V \iff U \vdash_{CSP(SST)} V.$$

Д о к а з а т е л ь с т в о. Предположим, что $U \vdash_{SST} V$. Тогда существует цепочка элементарных преобразований

$$U = U_0 \rightarrow U_1 \rightarrow \dots \rightarrow U_k \rightarrow U_{k+1} = V$$

полусистемы Туэ SST . Докажем, что при любом t ($0 \leq t \leq k$)

$$U_t \vdash_{CSP(SST)} U_{t+1}.$$

Если переход $U_t \rightarrow U_{t+1}$ в полусистеме Туэ SST имел вид

$$U_t = X A_i Y \rightarrow X B_i Y = U_{t+1},$$

то в канонической системе Поста $CSP(SST)$ получаем

$$U_t = X A_i Y \vdash_{CSP(SST)} A_i Y X' \vdash_{CSP(SST)} Y X' B'_i \vdash_{CSP(SST)} \\ X' B'_i Y' \vdash_{CSP(SST)} B'_i Y' X \vdash_{CSP(SST)} \\ Y' X B_i \vdash_{CSP(SST)} X B_i Y = U_{t+1}.$$

Теперь предположим, что $U \vdash_{CSP(SST)} V$. Тогда существует цепочка элементарных преобразований

$$U = U_0 \rightarrow U_1 \rightarrow \dots \rightarrow U_k \rightarrow U_{k+1} = V$$

канонической системы Поста $CSP(SST)$.

Индукцией по t ($0 \leq t \leq k$) легко доказать, что каждое слово U_t имеет вид либо CD' , либо $D'C$.

Определим преобразование $*$ равенствами

$$(DC')^* = (D'C)^* = CD.$$

Рассматривая все возможные случаи применения к словам вида CD' и $D'C$ продукций канонической системы Поста $CSP(SST)$, легко убедиться, что при любом t ($0 \leq t \leq k$) либо $(U_{t+1})^* = (U_t)^*$, либо $(U_{t+1})^*$ получается из $(U_t)^*$ применением одного элементарного преобразования полусистемы Туэ SST .

Так как U и V – слова в алфавите полусистемы Туэ SST , то $U^* = U$ и $V^* = V$, поэтому $U \vdash_{SST} V$. \square

Взяв в качестве SST такую полусистему Туэ, что для некоторого слова W_0 в ее алфавите множество всех выводимых из него слов не является рекурсивным, получим следующее следствие.

Следствие. *Существует каноническая система Поста $CSP(SST)$ такая, что для некоторого слова W_0 в ее алфавите множество всех выводимых из него слов не является рекурсивным.*

В теории алгоритмически неразрешимых проблем важную роль играет следующая проблема.

Проблема соответствия Поста. *По произвольному конечному множеству пар слов*

$$\{ \langle A_i, B_i \rangle \mid i \in I \}$$

в некотором алфавите $\mathcal{A} = \{a_1, \dots, a_n\}$ требуется определить, существует ли такая последовательность индексов i_1, \dots, i_m , что графически равны слова

$$A_{i_1} \dots A_{i_m} \quad \text{и} \quad B_{i_1} \dots B_{i_m}.$$

Будем говорить, что множество пар слов

$$\{ \langle A_i, B_i \rangle \mid i \in I \}$$

образует систему соответствия. Если существует требуемая последовательность индексов i_1, \dots, i_m , то соответствующее слово

$$A_{i_1} \dots A_{i_m} = B_{i_1} \dots B_{i_m}$$

называется решением системы соответствия.

Еще в 1946 году [125] Э. Пост доказал алгоритмическую неразрешимость этой чисто комбинаторной проблемы, получившей впоследствии название **проблема соответствия Поста**. Первоначальное доказательство Э. Поста было достаточно сложным. Позже оно было значительно упрощено. Мы приведем предложенное Флойдом достаточно простое доказательство этого весьма важного результата, так как он широко используется, например, в доказательствах неразрешимости ряда алгоритмических проблем для формальных грамматик.

Пусть SST – произвольная полусистема Туэ с алфавитом

$$\mathcal{A} = \{a_1, \dots, a_n\}$$

и с системой продукций

$$g_1 \rightarrow h_1, \dots, g_k \rightarrow h_k.$$

Рассмотрим новый алфавит

$$\mathcal{B} = \{a_1, \dots, a_n, a'_1, \dots, a'_n, [,], *, *'\}.$$

Для произвольного слова U в алфавите \mathcal{A} обозначим через U' слово, полученное из слова U одновременной заменой букв a_1, \dots, a_n на a'_1, \dots, a'_n .

Считаем, что система продукций полусистемы Туэ SST включает все "тождественные" продукции

$$a_1 \rightarrow a_1, \dots, a_n \rightarrow a_n.$$

По произвольным двум словам U и V в алфавите \mathcal{A} построим систему соответствия Поста $SC(U, V)$

$$\begin{aligned} \langle h_1, g'_1 \rangle, \dots, \langle h_k, g'_k \rangle, \\ \langle h'_1, g_1 \rangle, \dots, \langle h'_k, g_k \rangle, \\ \langle [U*, [, \langle *, *' \rangle, \langle *', * \rangle, \langle], *'V] \rangle. \end{aligned}$$

Теорема 2.3. Для произвольных слов U и V в алфавите $\mathcal{A} = \{a_1, \dots, a_n\}$ полусистемы Туэ SST справедлива эквивалентность

$$U \vdash_{SST}^* V \iff \text{система соответствия } SC(U, V) \text{ имеет решение.}$$

Доказательство. Предположим, что

$$U \vdash_{SST}^* V.$$

Тогда существует цепочка элементарных преобразований полусистемы Туэ SST , ведущая от слова U к слову V ,

$$U = U_1 \rightarrow U_2 \rightarrow \dots \rightarrow U_{m-1} \rightarrow U_m = V.$$

Так как по предположению в число продукций полусистемы Туэ SST входят все "тождественные" продукции $a_i \rightarrow a_i$ ($i = 1, \dots, n$), то можно считать, что m – нечетное число.

Покажем, что слово

$$[U_1 * U'_2 *' U_3 * \dots * U'_{m-1} *' U_m]$$

является решением системы соответствия Поста $SC(U, V)$.

Рассмотрим следующие два разбиения на подслова этого слова

$$\begin{aligned} [U_1 * | U'_2 *' | U_3 * | \dots * | U'_{m-1} *' | U_m |] = \\ [| U_1 * | U'_2 *' | U_3 * | \dots * | U'_{m-1} | *' U_m]. \end{aligned}$$

Соответствие слов $[U_1 * \text{ и } [, \text{ слов }] \text{ и } *' U_m]$, слов $*'$ и $*$ и слов $*$ и $*'$ очевидно.

Остается установить соответствие при четном t слов U'_t и U_{t-1} , а при нечетном t – слов U_t и U'_{t-1} .

Так как переход $U_{t-1} \rightarrow U_t$ является элементарным преобразованием, то для подходящих слов L и R в алфавите \mathcal{A} при некотором j ($1 \leq j \leq k$) выполняются равенства

$$U_{t-1} = L g_j R, \quad U_t = L h_j R.$$

Значит верны равенства

$$U'_t = L' h'_j R', \quad U_{t-1} = L g_j R$$

$$U_t = L h_j R, \quad U'_{t-1} = L' g'_j R'.$$

”Тождественные” продукции $a_i \rightarrow a_i$ ($i = 1, \dots, n$) дают соответствия слов a_i и a'_i , a'_i и a_i , а значит, и слов L и L' , R и R' , L' и L , R' и R . Это вместе с соответствием слов h_j и g'_j , h'_j и g_j дает соответствие слов U_t и U'_{t-1} , U'_t и U_{t-1} .

Значит указанное слово действительно является решением системы соответствия Поста $SC(U, V)$.

Для доказательства обратного предположим, что система соответствия Поста $SC(U, V)$ имеет решение и слово W является самым коротким решением.

Анализ системы соответствия Поста $SC(U, V)$ показывает, что если S и R – слова в алфавите \mathcal{A} и слову S' соответствует слово R , то слово S может быть получено из слова R конечным числом преобразований полусистемы Туэ SST , а если слову S соответствует слово R' , то слово S может быть получено из слова R конечным числом преобразований полусистемы Туэ SST .

Нетрудно понять, что слово W должно начинаться символом $[$, а оканчиваться символом $]$.

Так как по предположению слово W является самым коротким решением, то нетрудно понять, символы $[$ и $]$ входят в слово W лишь по одному разу. Значит, слово W имеет вид

$$[U * \dots *' V],$$

и решение для системы соответствия Поста $SC(U, V)$ дается следующими двумя разбиениями этого слова

$$W = [U * | \dots *' V|], \quad W = [|U * \dots | *' V].$$

Заметим, самая левая $*$ второго разбиения должна соответствовать некоторой $*'$ первого разбиения, а самой правой $*'$ первого разбиения соответствует некоторая $*$ второго разбиения.

Значит, в первом разбиении после первой $*$ идет некоторое слово S' , которому соответствует слово U , поэтому

$$U \vdash_{SST}^* S'.$$

Рассмотрим вхождение слова S' во второе разбиение. Оно соответствует некоторому слову R из первого разбиения и

$$S \vdash_{SST}^* R.$$

Поэтому

$$U \vdash_{ST}^* R.$$

Продолжая рассуждение аналогичным образом, получим

$$U \vdash_{ST}^* V.$$

□

Так как проблема выводимости для полусистем Туэ алгоритмически неразрешима, то получаем в качестве непосредственного следствия доказанной теоремы следующее утверждение

Следствие. *Проблема соответствия Поста алгоритмически неразрешима, т.е. невозможен алгоритм, позволяющий по произвольному конечному множеству пар слов*

$$\{ \langle A_i, B_i \rangle \mid i \in I \}$$

в произвольном алфавите $\mathcal{A} = \{a_1, \dots, a_n\}$ определить, существует ли такая последовательность индексов i_1, \dots, i_m , что графически равны слова

$$A_{i_1} \dots A_{i_m} \quad \text{и} \quad B_{i_1} \dots B_{i_m}.$$

Проблема соответствия Поста допускает следующую алгебраическую формулировку.

Обозначим через Π_k свободную полугруппу со свободными образующими a_1, \dots, a_k . Любой гомоморфизм φ полугруппы Π_m в полугруппу Π_n задается своими значениями $\varphi(a_1), \dots, \varphi(a_m)$ на свободных образующих a_1, \dots, a_k . Для двух гомоморфизмов φ и ψ полугруппы Π_m в полугруппу Π_n определяется их *множество совпадения*

$$E(\varphi, \psi) = \{ w \mid w \in \Pi_m \& \varphi(w) = \psi(w) \}.$$

Множество $E(\varphi, \psi)$ содержит пустое слово.

Проблема соответствия Поста – это вопрос о том, содержит ли множество $E(\varphi, \psi)$ непустое слово.

Любой гомоморфизм φ полугруппы Π_m в полугруппу Π_n задает некоторое *кодирование (алфавитное кодирование)* слов в алфавите Π_m словами в алфавите Π_n .

В такой постановке проблема соответствия Поста – это вопрос о том, *можно ли по двум кодированиям φ и ψ слов в алфавите Π_m словами в алфавите Π_n определить, найдется ли такое слово W в алфавите Π_m , которое φ и ψ кодируют одинаково, т.е. одним и тем же словом (кодом) $\varphi(W) = \psi(W)$.*

Рассмотрим некоторые алгоритмические проблемы, возникающие при изучении теории формальных языков и формальных грамматик.

Напомним, что формальная грамматика Γ задается алфавитом $\mathcal{A} = N \cup T$, разбитым в объединение двух непересекающихся подмножеств N и T , называемых соответственно множеством *нетерминальных* и множеством *терминальных* символов, системой продукций Π и начальным символом $S \in N$. Получаем полусистему Туэ

$$SST(\Gamma) = \langle \mathcal{A} \mid \Pi \rangle.$$

Специфика формальных грамматик проявляется в задании ими языков.

Формальная грамматика Γ определяет язык $L(\Gamma)$, состоящий из всех слов в алфавите T терминальных символов, выводимых из слова S , т.е.

$$L(\Gamma) = \{ W \mid W \in T^* \text{ \& } S \vdash_{SST(\Gamma)}^* W \}.$$

Классификация формальных грамматик ведется по виду их продукций. Важный класс составляют *контекстно свободные грамматики*, все продукции которых имеют вид $\alpha \rightarrow h$, где α – нетерминальный символ.

Для формальных грамматик естественно формулируются алгоритмические проблемы, например, проблема вхождения в язык $L(\Gamma)$. В общем случае она алгоритмически неразрешима, но для контекстно свободных грамматик разрешима.

Рассмотрим для формальных грамматик *проблему пустоты пересечения задаваемых ими языков*.

Проблема пустоты пересечения языков. *По произвольным двум формальным грамматикам Γ_1 и Γ_2 требуется определить, пусто ли пересечение $L(\Gamma_1) \cap L(\Gamma_2)$ задаваемых этими грамматиками языков.*

Теорема 2.4. *Невозможен алгоритм, позволяющий по произвольным двум контекстно свободным грамматикам Γ_1 и Γ_2 определить, пусто ли пересечение $L(\Gamma_1) \cap L(\Gamma_2)$ задаваемых этими грамматиками языков.*

Д о к а з а т е л ь с т в о. Рассмотрим произвольную систему соответствия Поста в алфавите $\mathcal{A} = \{a_1, \dots, a_n\}$, заданную конечным множеством пар слов

$$\{ \langle A_i, B_i \rangle \mid 1 \leq i \leq m \}.$$

Построим пару контекстно свободных грамматик Γ_1 и Γ_2 , у которых одно и то же множество $T = \mathcal{A} \cup \{c_1, \dots, c_m\}$ терминальных символов. У грамматики Γ_1 множество нетерминальных символов состоит лишь из S_1 , а у Γ_2 – лишь из S_2 .

Множество Π_1 продукций грамматики Γ_1 состоит из всех продукций вида $S_1 \rightarrow A_i S_1 c_i$ и $S_1 \rightarrow A_i c_i$ ($i = 1, \dots, m$).

Множество Π_2 продукций грамматики Γ_2 состоит из всех продукций вида $S_2 \rightarrow B_i S_2 c_i$ и $S_2 \rightarrow B_i c_i$ ($i = 1, \dots, m$).

Тогда

$$\begin{aligned} L(\Gamma_1) &= \{ A_{i_1} A_{i_2} \dots A_{i_t} c_{i_t} \dots c_{i_2} c_{i_1} \}, \\ L(\Gamma_2) &= \{ B_{j_1} B_{j_2} \dots B_{j_k} c_{j_k} \dots c_{j_2} c_{j_1} \}. \end{aligned}$$

Поэтому исходная система соответствия Поста имеет решение тогда и только тогда, когда для построенных по ней контекстно свободных грамматик Γ_1 и Γ_2 $L(\Gamma_1) \cap L(\Gamma_2) \neq \emptyset$. Для завершения доказательства теоремы достаточно сослаться на установленную алгоритмическую неразрешимость проблемы соответствия Поста. \square

В качестве еще одного примера алгоритмически неразрешимой проблемы для формальных грамматик и языков рассмотрим *проблему двусмысленности*.

Определение 2.1. *Формальная грамматика Γ с начальным символом S называется **двусмысленной**, если для некоторого слова W из ее языка $L(\Gamma)$ существуют два различных вывода, т.е. такие две цепочки элементарных преобразований*

$$S \equiv U_1 \rightarrow U_2 \rightarrow \dots \rightarrow U_{m-1} \rightarrow U_m \equiv W$$

и

$$S \equiv V_1 \rightarrow V_2 \rightarrow \dots \rightarrow V_{k-1} \rightarrow V_k \equiv V,$$

что

$$\langle U_1, U_2, \dots, U_m \rangle \neq \langle V_1, V_2, \dots, V_k \rangle.$$

Проблема двусмысленности грамматики По произвольной грамматике требуется определить, является ли она двусмысленной.

Теорема 2.5. *Невозможен алгоритм, позволяющий по произвольной контекстно свободной грамматике Γ определить, является ли она двусмысленной.*

До к а з а т е л ь с т в о. Как в доказательстве предыдущей теоремы по произвольной системе соответствия Поста SC строим пару контекстно свободных грамматик Γ_1 и Γ_2 .

По грамматикам Γ_1 и Γ_2 строим грамматику Γ , алфавит которой получается объединением их алфавитов и добавлением нового нетерминального символа S . Множество продукций грамматики Γ получается добавлением к объединению множеств продукций грамматик Γ_1 и Γ_2 двух продукций

$$S \rightarrow S_1, \quad S \rightarrow S_2.$$

Нетрудно понять, что $L(\Gamma) = L(\Gamma_1) \cup L(\Gamma_2)$.

Ясно, что грамматики Γ_1 и Γ_2 не являются двусмысленными. Поэтому *грамматика Γ является двусмысленной тогда и только тогда, когда $L(\Gamma_1) \cap L(\Gamma_2) \neq \emptyset$.*

И для завершения доказательства теоремы достаточно либо сослаться на предыдущую теорему, либо заметить, что

исходная система соответствия Поста SC имеет решение тогда и только тогда, грамматика Γ является двусмысленной. \square

Разнообразные примеры алгоритмически неразрешимых проблем для формальных грамматик и языков содержатся в известных монографиях А.В. Гладкого [13] С. Гинзбурга [12].

В заключение приведем доказательство теоремы А.А. Маркова – Э. Поста о неразрешимости проблемы равенства для полугрупп.

Как уже отмечалось выше, в 1914 году в работе [130] норвежский математик Аксель Туэ предложил универсальный способ задания полугрупп с помощью порождающих элементов и определяющих соотношений. Там же он впервые сформулировал алгоритмическую *проблему распознавания равенства слов для конечно определенных полугрупп*, т.е. таких заданий, где число порождающих и число определяющих соотношений конечно. Интересно отметить, что еще в 1882–1883 гг. Вальтер фон Дик ввел аналогичный способ задания групп – *задание групп образующими элементами и определяющими соотношениями*. Понятие группы, в некотором смысле, является более классическим, чем понятие полугруппы, а тем более, чем понятие системы Туэ.

Напомним некоторые необходимые определения. Множество \mathcal{A}^* всех слов в алфавите \mathcal{A} образует свободную полугруппу относительно бинарной операции приписывания одного слова к другому. При этом пустое слово играет роль единицы и обозначается через 1.

Рассмотрим произвольный алфавит \mathcal{A} и некоторое множество \mathcal{S} упорядоченных пар слов в этом алфавите \mathcal{A} . По сложившейся традиции упорядоченную пару слов $\langle A, B \rangle$ при задании полугруппы будем записывать в виде равенства $A = B$. Эти равенства, называемые *определяющими соотношениями*, определяют на множестве всех слов некоторое *отношение эквивалентности*, факторизация по которому переводит свободную полугруппу \mathcal{A}^* в полугруппу, заданную определяющими соотношениями множества \mathcal{S} и обозначаемую через

$$\Pi(\mathcal{A}, \mathcal{S}) = \langle \mathcal{A} \mid \mathcal{S} \rangle.$$

Отношение равенства слов в таком образом заданной полугруппе $\Pi(\mathcal{A}, \mathcal{S})$ определяется следующим образом.

Левым элементарным преобразованием, отвечающим определяющему соотношению $A = B$, называется любой переход вида

$$UAV \rightarrow UB V,$$

где U и V – произвольные слова в алфавите \mathcal{A} .

Правыми элементарными преобразованиями называются переходы вида

$$UBV \rightarrow UAV.$$

В дальнейшем запись $P \rightarrow Q$ будет обозначать, что слово Q получается из слова P с помощью одного элементарного преобразования рассматриваемой полугруппы.

Два слова P и Q в алфавите \mathcal{A} называются *эквивалентными в полугруппе* $\Pi(\mathcal{S})$, если либо $P = Q$, либо существует цепочка элементарных преобразований вида

$$P = P_0 \rightarrow P_1 \rightarrow \dots \rightarrow P_k = Q$$

(записывается $P = Q$ в $\Pi(\mathcal{S})$).

Очевидно, определенное таким образом отношение эквивалентности *согласовано* с операцией умножения слов, т.е. если $P_1 \equiv Q_1$ и $P_2 \equiv Q_2$, то $P_1 P_2 \equiv Q_1 Q_2$.

Элементами полугруппы $\Pi(\mathcal{A}, \mathcal{S})$ являются классы эквивалентности $[X]$, состоящие из всех слов, эквивалентных слову X в определенном выше смысле. Полугруппа $\Pi(\mathcal{A}, \mathcal{S})$ представляет собой множество всех классов эквивалентности с ассоциативной операцией, определенной равенством $[X] \cdot [Y] = [XY]$. Нетрудно понять, что это определение корректно, т.е. результат не зависит от выбора представителей X и Y классов эквивалентности.

Очевидно, если добавить к множеству \mathcal{A} новую образующую b , а к соотношениям равенство вида $b = W$, где W – слово в алфавите \mathcal{A} , то получится новое задание той же полугруппы. Также очевидно, что два задания полугрупп $\Pi_1(\mathcal{A}, \mathcal{S}_1)$ и $\Pi_2(\mathcal{A}, \mathcal{S}_2)$ определяют одну и ту же полугруппу, если они удовлетворяют условию:

любое равенство из \mathcal{S}_2 выполнено (выводимо) в Π_1 и любое равенство из \mathcal{S}_1 выполнено (выводимо) в Π_2 .

Если множества \mathcal{A} и \mathcal{S} конечны, то задание полугруппы $\Pi(\mathcal{A}, \mathcal{S})$ может быть записано в виде

$$\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle.$$

Очевидно, что, для того чтобы полугруппа с единицей была группой, достаточно чтобы в ней имелись обратные элементы для всех порождающих. Задание группы с помощью образующих и определяющих соотношений осуществляется следующим образом.

К данному исходному алфавиту

$$\mathcal{A} = \{a_1, a_2, \dots, a_n, \dots\} \quad (3)$$

добавляется алфавит "двойников", называемых обратными степенями образующих,

$$\mathcal{A}^{-1} = \{a_1^{-1}, a_2^{-1}, \dots, a_i^{-1}, \dots\}.$$

Полученное объединение называется *групповым алфавитом*. Для обеспечения того, чтобы буквы a_i^{-1} были обратными для букв a_i при задании любой группы с образующими (3) используются так называемые *тривиальные определяющие соотношения*, имеющие вид

$$a_i^{-1} a_i = 1, \quad a_i a_i^{-1} = 1. \quad (4)$$

Для задания группы к тривиальным соотношениям (4) добавляются определяющие соотношения в групповом алфавите, которые в данном случае всегда

можно записать в виде равенств

$$A_i = 1 \quad (i = 1, 2, \dots, m). \quad (5)$$

Так как добавление обратных степеней образующих и тривиальных соотношений однозначно определяется заданием множества порождающих (3), то обычно говорят о группе G , заданной порождающими (3) и определяющими соотношениями (5), которая обозначается через

$$\langle\langle a_1, a_2, \dots, a_n \mid A_1 = 1, A_2 = 1, \dots, A_m = 1 \rangle\rangle.$$

Соответствующие тривиальным соотношениям (4) элементарные преобразования

$$Ua_i^{-1}a_iV \rightarrow UV \quad \text{и} \quad Ua_ia_i^{-1}V \rightarrow UV \quad (6)$$

называются *сокращениями* букв a_i и a_i^{-1} , а обратные им преобразования – *вставками* этих букв.

Слово X называется *несократимым*, если оно не содержит вхождений подслов вида $a_ia_i^{-1}$ или вида $a_i^{-1}a_i$. Очевидно, в любой группе произвольное слово можно преобразовать в равное ему несократимое слово, используя только преобразования (6).

Напомним, что элементами группы G являются классы эквивалентных слов в групповом алфавите, которые однозначно определяются любым своим представителем.

Если

$$X = a_{i_1}^{\sigma_1} a_{i_2}^{\sigma_2} \dots a_{i_{k-1}}^{\sigma_{k-1}} a_{i_k}^{\sigma_k},$$

где $\sigma_j = \pm 1$ ($j = 1, 2, \dots, k$), то *слово*

$$X^{-1} = a_{i_k}^{-\sigma_k} a_{i_{k-1}}^{-\sigma_{k-1}} \dots a_{i_2}^{-\sigma_2} a_{i_1}^{-\sigma_1}$$

называется *обратным к слову X* . Очевидно, слова X и X^{-1} определяют взаимно обратные элементы группы.

Если множество определяющих соотношений (5) пусто, то мы получаем *свободную группу* с порождающими (3). Она обозначается через F_n . Число n называется ее *рангом*.

Проблема распознавания равенства слов для полугрупп, сформулированная А. Туэ в 1914 году [130], состоит в следующем.

Построить алгоритм, позволяющий для произвольной полугруппы Π и любых двух слов U и V в ее алфавите определить, представляют ли они один и тот же элемент этой полугруппы, т.е. эквивалентны ли они в Π .

Проблема равенства для конечно определенных групп была сформулирована М. Дэном двумя годами ранее в работе [104]. Ее иногда еще называют *проблемой тождества*.

Проблему равенства можно ставить и для каждой полугруппы или группы в отдельности. При этом мы получим *частную проблему равенства* для фиксированной полугруппы (группы).

Конечно, М. Дэн и А. Туэ не предполагали, что искомый алгоритм может не существовать. Они просто ставили задачу построения алгоритма для решения важной проблемы, связанной с универсальным способом задания полугрупп и групп. Более того, появившиеся только в середине XX века доказательства неразрешимости этих проблем оказались неожиданными для широкого круга математиков.

В 1947 году А.А. Марков [39] и Э. Пост [125] независимо доказали *существование конечно определенных полугрупп с неразрешимой проблемой равенства слов*. Это был первый в истории математики пример неразрешимой проблемы, которая была сформулирована задолго до появления в математике точного определения понятия алгоритма. Приведем изложенное в работе [3] доказательство этого результата, существенно более простое, чем оригинальные доказательства самих авторов. Оно основано на доказанной выше неразрешимости проблемы остановки для машины Тьюринга T_0 .

Рассмотрим произвольную машину Тьюринга T с внешним алфавитом $A_T = \{a_0, a_1, \dots, a_n\}$, алфавитом внутренних состояний $Q_T = \{q_0, q_1, \dots, q_m\}$ и программой P_T . По этой программе P_T естественным образом построим конечно определенную полугруппу $S(T)$, заданную множеством образующих элементов

$$\{a_0, a_1, \dots, a_n, q_0, q_1, \dots, q_m, h\}$$

и некоторым множеством определяющих соотношений $\mathcal{R}(S(T))$, которая строится по программе P_T .

Для каждой команды $P_T(i, j)$ из P_T включаем в $\mathcal{R}(S(T))$ соотношения определенного ниже вида.

I. Если $P_T(i, j)$ имеет вид $q_i a_j \rightarrow q_r a_t$, то включаем в $\mathcal{R}(S(T))$ одно соотношение $q_i a_j = q_r a_t$.

II. Если команда $P_T(i, j)$ имеет вид $q_i a_j \rightarrow q_r L$, то включаем все соотношения вида $a_t q_i a_j = q_r a_t a_j$ при $t = 0, \dots, n$ и еще одно соотношение $h q_i a_j = h q_r a_0 a_j$.

III. Если команда $P_T(i, j)$ имеет вид $q_i a_j \rightarrow q_r R$, то включаем все соотношения вида $q_i a_t a_j = a_t q_r a_j$ при $t = 0, \dots, n$ и еще одно соотношение $q_i a_j h = a_j q_r a_0 h$.

IV. Кроме того, добавим в $\mathcal{R}(S(T))$ еще $h q_0 h = q_0$ и все соотношения вида $q_0 a_t = q_0$ и $a_t q_0 = q_0$.

Каждой конфигурации $X = E_1 q_i E_2$ машины T ставим в соответствие слово $h E_1 q_i E_2 h$, которое будем называть *словом Поста*.

Заметим, что элементарные преобразования типов I, II, III любое слово Поста переводят в слово Поста. Следующая лемма очевидна.

Лемма 2.3. *Машина Тьюринга T переводит за один такт работы конфигурацию X в конфигурацию Y в том и только том случае, если слово Поста $h X h$ переводится в слово Поста $h Y h$ одним левым элементарным преобразованием, соответствующим команде, применяемой машиной T .*

Основной является следующая лемма.

Лемма 2.4. *Машина Тьюринга T , начав работать в конфигурации X , через конечное число тактов работы остановится тогда и только тогда, когда в полугруппе $S(T)$ выполняется равенство $hXh = q_0$.*

Д о к а з а т е л ь с т в о. Пусть машина Тьюринга T , начав работать в конфигурации X , через конечное число тактов работы остановится, т.е. имеется конечная цепочка конфигураций $X, X_1, X_2, \dots, X_{k-1}, X_k$, оканчивающаяся заключительной конфигурацией X_k . Тогда по лемме 2.3 имеем последовательность элементарных преобразований полугруппы $S(T)$

$$hXh \rightarrow hX_1h \rightarrow hX_2h \rightarrow \dots \rightarrow hX_{k-1}h \rightarrow hX_kh,$$

где hX_kh имеет вид $hZ_1q_0Z_2h$. Используя определяющие соотношения из IV-й группы, получаем, что в полугруппе $S(T)$ выполняется равенство $hXh = q_0$.

Предположим теперь, что в полугруппе $S(T)$ выполняется равенство $hXh = q_0$. Это значит, что существует цепочка элементарных преобразований

$$hXh = Y_0 \rightarrow Y_1 \rightarrow Y_2 \rightarrow \dots \rightarrow Y_{k-1} \rightarrow Y_k = q_0. \quad (1)$$

Очевидно, в каждое слово Y_j из последовательности (1) входит только одна q -буква. Рассмотрим кратчайшую цепочку элементарных преобразований вида (1), заканчивающуюся словом, содержащим q_0 :

$$hXh = Y_0 \rightarrow Y_1 \rightarrow Y_2 \rightarrow \dots \rightarrow Y_{r-1} \rightarrow Y_r = Z_1q_0Z_2. \quad (2)$$

В преобразованиях (2) определяющие соотношения полугруппы $S(T)$ из IV-й группы не используются. Поэтому в (2) все слова являются словами Поста.

Покажем, что в цепочке (2) все преобразования левые. Допустим, что это не так. Пусть

$$Y_t \rightarrow Y_{t+1} \quad (3)$$

есть самое последнее правое элементарное преобразование в (2). Тогда $t + 1 < r$ и за преобразованием (3) должно следовать левое преобразование

$$Y_{t+1} \rightarrow Y_{t+2}. \quad (4)$$

Так как среди левых частей определяющих соотношений типов I, II, III есть только одно слово E , содержащее данное двубуквенное подслово q_ia_j , то в (4) может использоваться лишь то же самое соотношение, что и в (3). Более того, однозначно определено также и вхождение всего определяющего слова E в слово Y_{t+1} . Поэтому получаем $Y_{t+2} = Y_t$, что противоречит условию минимальности последовательности (2). Значит, (2) состоит только из левых элементарных преобразований. Тогда по лемме 2.3 машина T из конфигурации X через конечное число шагов перейдет в заключительное состояние q_0 и остановится. Лемма 2.4 доказана. \square

Пусть T_0 – машина Тьюринга с неразрешимой проблемой остановки. Тогда из леммы 2.4 следует

Теорема 2.6 (Марков – Пост). *Невозможно создать алгоритм, позволяющий для произвольного слова X в алфавите образующих полугруппы $S(T_0)$ ответить на вопрос, равно ли оно q_0 в полугруппе $S(T_0)$.*

Заметим, что любая конечно определенная полугруппа вкладывается в полугруппу, заданную двумя порождающими элементами и конечным числом определяющих соотношений. Для произвольного слова W в алфавите $\{a_1, \dots, a_n\}$ обозначим через W^* слово в двубуквенном алфавите $\{a, b\}$, полученное из слова W заменой каждой буквы a_i на слово $aba^{i+1}b^{i+1}$. Легко проверить, что отображение

$$\varphi : W \rightarrow W^*$$

является изоморфным вложением полугруппы

$$\Pi = \langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle.$$

в полугруппу

$$\Pi^* = \langle a, b \mid A_1^* = B_1^*, \dots, A_m^* = B_m^* \rangle,$$

т.е. для любых двух слов W в алфавите $\{a_1, \dots, a_n\}$ верна эквивалентность

$$U \stackrel{\Pi}{=} V \iff U^* \stackrel{\Pi^*}{=} V^*.$$

Поэтому если для полугруппы Π проблема равенства слов неразрешима, то она неразрешима и для полугруппы Π^* . Значит *существует полугруппа с неразрешимой проблемой равенства, заданная двумя порождающими и конечным числом определяющих соотношений.*

Наиболее простые (по общей длине задания) примеры полугрупп с неразрешимой проблемой распознавания равенства слов были построены в 1956 году независимо Г.С. Цейтиным [91] и Д. Скоттом [136]. Оба эти задания содержали 7 определяющих соотношений. Приводим пример Цейтина.

Теорема 2.7 (Г.С.Цейтин [91]). *Полугруппа Π , заданная образующими a, b, c, d, e и определяющими соотношениями*

$$ac = ca, ad = da, bc = cb, bd = db, ce = eca, de = edb, csa = csaе,$$

имеет неразрешимую проблему равенства слов.

Полугруппа Г.С. Цейтина неоднократно использовалась при построении других простых примеров алгебраических систем с неразрешимой проблемой равенства.

В 1966 году Г.С. Маканин построил вложение полугруппы Г.С. Цейтина в полугруппу, заданную следующими пятью определяющими соотношениями

$$\begin{aligned} gggf &= ffgg, fgggff = gffffg, fgggffffgg = ffggffffgге, \\ eggff &= ffe, efgggff = gffe, \end{aligned}$$

которая поэтому также имеет неразрешимую проблему равенства слов.

В том же году Ю.В. Матиясевич построил общую конструкцию вложения произвольной конечно определенной полугруппы в полугруппу с пятью определяющими соотношениями. Вскоре он усилил этот результат [47], построив примеры полугрупп с тремя определяющими соотношениями и неразрешимой проблемой равенства.

В качестве простого следствия доказанных теорем получим доказательство алгоритмической неразрешимости проблемы изоморфизма для конечно определенных полугрупп.

Определение 2.2. Полугруппа Π называется хопфовой, если каждый ее сюръективный гомоморфизм на себя является биекцией, т.е. автоморфизмом.

Лемма 2.5. Если в задании

$$\langle a_1, \dots, a_n \mid A_1 = B_1, \dots, A_m = B_m \rangle$$

полугруппы Π длина каждого из слов A_i и B_i ($i = 1, \dots, m$) не меньше двух, то Π – хопфова полугруппа.

Доказательство. Пусть φ – сюръективный гомоморфизм полугруппы Π на себя. Для образующего элемента a_j найдется такое слово $V_j(a_1, \dots, a_n)$, что

$$a_j \underset{\Pi}{=} \varphi(V_j(a_1, \dots, a_n)) \underset{\Pi}{=} V_j(\varphi(a_1), \dots, \varphi(a_n)).$$

Так как длина каждого из слов A_i и B_i ($i = 1, \dots, m$) не меньше двух, то V_j – это буква. Поэтому отображение φ индуцирует на множестве образующих $\{a_1, \dots, a_n\}$ подстановку σ_φ . Значит, $\sigma_\varphi^{n!} = id$. Поэтому и $\varphi^{n!} = id$, значит, φ – биекция. \square

Заменим в каждом определяющем соотношении из множества определяющих соотношений $\mathcal{R}(S(T))$ построенной выше по произвольной машине Тьюринга T полугруппы $S(T)$ всюду букву q_0 на q_0q_0 сохранив все остальное. Полученную полугруппу обозначим через $S^+(T)$.

Данное выше доказательство леммы 2.4, как легко понять, доказывает и следующую лемму.

Лемма 2.6. Машина Тьюринга T , начав работать в конфигурации X , через конечное число тактов работы остановится тогда и только тогда, когда в полугруппе $S^+(T)$ выполняется равенство $hXh = q_0q_0$.

Поэтому если для машины Тьюринга T неразрешима проблема остановки, то для полугруппы $S^+(T)$ неразрешима проблема равенства слов.

Для произвольных двух слов U и V в алфавите образующих полугруппы $S^+(T)$ обозначим через $S^+(T)_{U,V}$ полугруппу, задание которой получается из

задания полугруппы $S^+(T)$ добавлением к ее определяющим соотношениям нового соотношения $U = V$. Лемма 2.6 дает эквивалентность

$$U \underset{S^-(T)}{=} V \iff \text{полугруппы } S^+(T)_{U,V} \text{ и } S^-(T) \text{ изоморфны.}$$

Это дает доказательство следующей теоремы

Теорема 2.8. *Невозможно создать алгоритм, позволяющий для произвольной полугруппы $S^+(T)_{U,V}$ определить, изоморфна ли она полугруппе $S^+(T)$.*

Нетрудно понять, что для произвольной конфигурации X , т.е. слова вида Aq_iB , где A и B – слова в алфавите образующих полугруппы $S^+(T)$, причем слово B непустое, в слово AB не входит q -символ и $i \neq 0$, справедлива эквивалентность

элемент $[hUh]$ входит в подполугруппу, порожденную элементом $[q_0q_0]$, тогда и только тогда, когда $hUh \underset{S^+(T)}{=} q_0q_0$.

Поэтому для полугруппы $S^+(T)$ алгоритмически неразрешима проблема вхождения в циклические подполугруппы.

§3. Недетерминированные многоленточные машины Тьюринга

В предыдущих параграфах рассматривался в определенном смысле простейший вариант машины Тьюринга – *детерминированные машины Тьюринга с одной конечной лентой потенциально бесконечной в обе стороны*. Как уже отмечалось выше, существуют различные варианты машин Тьюринга – с одной или с несколькими лентами, с дополнительной входной лентой, с входной, выходной и рабочей лентами, с многомерными лентами и т.д., детерминированные и недетерминированные. *С точки зрения принципиальных вычислительных возможностей различные варианты машин Тьюринга эквивалентны*. Однако особенности вычислительной модели играют существенную роль, когда нас интересует не только принципиальная возможность реализации того или иного алгоритма, но и сложностные характеристики его выполнения на рассматриваемой модели.

Рассматривавшиеся выше одноленточные машины Тьюринга являются **детерминированными** вычислительными устройствами – *на каждом такте работы машина Тьюринга может выполнить лишь одну команду из своей программы, поэтому из текущего незаключительного состояния машина Тьюринга переходит в однозначно определенное следующее состояние*.

Достаточно естественным обобщением понятия детерминированной машины Тьюринга является понятие **недетерминированной** машины Тьюринга, которая на каждом такте работы **может выполнить** одну из нескольких команд, что позволяет ей из текущего незаключительного состояния перейти в

одно из нескольких возможных следующих состояний, т.е. *следующее состояние не определяется однозначно текущим состоянием*.

В начале рассмотрим естественное обобщение одноленточной машины Тьюринга – k -ленточную машину Тьюринга.

k -ленточная машина Тьюринга T имеет *внешний алфавит*, или *алфавит ленточных символов*,

$$\mathcal{A}_T = \{a_0, a_1, \dots, a_n\},$$

внутренний алфавит, или *алфавит внутренних состояний*,

$$\mathcal{Q}_T = \{q_0, q_1, \dots, q_m\},$$

программу P_T .

Элементы внешнего алфавита называются просто *символами*, а элементы внутреннего алфавита – *внутренними состояниями*.

У k -ленточной машины Тьюринга имеется k *лент*, каждая из которых разбита на конечное число ячеек, в которые можно записывать символы из внешнего алфавита $\{a_0, a_1, \dots, a_n\}$, поэтому он и называется *алфавитом ленточных символов*. По чисто техническим причинам удобно считать, что среди ленточных символов имеется так называемый *пустой символ*, который традиционно обозначают через a_0 . Ячейки ленты упорядочены слева направо. Мы рассматриваем вариант определения машины Тьюринга, в котором предполагается, что ленты *потенциально бесконечны в обе стороны*. Это означает, что в процессе работы машины Тьюринга разрешается добавлять новые пустые ячейки как к правому, так и к левому концам любой ленты (считается, что в момент добавления в них записан пустой символ a_0). В принципе можно было бы считать ленты бесконечными, но потребовать, чтобы в каждый момент времени лишь конечное число их ячеек содержало непустые символы.

С каждой лентой связана своя *головка*, которая в каждый момент времени обозревает одну ячейку соответствующей ленты, может считывать обозреваемый символ и записывать в обозреваемую ячейку новый символ.

Можно считать, что у машины Тьюринга имеется управляющее устройство, которое в каждый момент времени находится в одном из *внутренних состояний* q_0, q_1, \dots, q_m . Состояние q_0 традиционно называется *заключительным состоянием*. Ради некоторой краткости мы часто будем говорить, что в текущий момент времени сама машина Тьюринга находится в одном из внутренних состояний, это не приведет к путанице.

Машина Тьюринга выполняет действия в дискретные моменты времени или, как говорят, *потактово*.

Работа машины Тьюринга состоит из отдельных тактов, выполняемых согласно ее программе P_T .

Программа P_T для k -ленточной (детерминированной) машины Тьюринга T – это конечный набор команд вида:

$$q_i \langle a_{j_1}, \dots, a_{j_k} \rangle \longrightarrow q_s \langle \langle a_{t_1}, D_1 \rangle, \dots, \langle a_{t_k}, D_k \rangle \rangle,$$

где $D_1, \dots, D_k \in \{L, S, R\}$.

При этом каждому набору $q_i \langle a_{j_1}, \dots, a_{j_k} \rangle$ при $i > 0$ соответствует ровно одна команда, в которой перед \longrightarrow стоит этот набор. Эту команду мы будем обозначать через

$$P_T(i, j_1, \dots, j_k).$$

Если в текущий момент времени детерминированная машина Тьюринга T находится в состоянии q_i при $i > 0$, а в обозреваемых ячейках лент находятся символы a_{j_1}, \dots, a_{j_k} , то машина T **должна выполнить** (выполняет) команду $P_T(i, j_1, \dots, j_k)$.

При этом, если эта команда имеет вид

$$q_i \langle a_{j_1}, \dots, a_{j_k} \rangle \longrightarrow q_s \langle \langle a_{t_1}, D_1 \rangle, \dots, \langle a_{t_k}, D_k \rangle \rangle,$$

то

- 1) машина переходит в состояние q_s ,
- 2) на l -ой ($1 \leq l \leq k$) ленте выполняется команда

$$q_i a_{j_l} \longrightarrow q_s a_{t_l} D_l,$$

т.е. в обозреваемой ячейке символ a_{j_l} заменяется на символ a_{t_l} и выполняется действие D_l как для одноленточной машины Тьюринга.

Если в результате выполнения некоторой команды машина *приходит в заключительное состояние* q_0 , то говорят, что она **останавливается**, так как по определению ни одна команда программы не начинается с q_0 .

В некоторых случаях программу P_T удобно рассматривать как отображение δ_T множества $(Q_T \setminus \{q_0\}) \times \mathcal{A}_T^k$ во множество $Q_T \times (\mathcal{A}_T \times \{L, S, R\})^k$

$$\delta_T : (Q_T \setminus \{q_0\}) \times \mathcal{A}_T^k \longrightarrow Q_T \times (\mathcal{A}_T \times \{L, S, R\})^k.$$

Полное описание состояния k -ленточной машины Тьюринга в данный момент времени можно, как и одноленточной машины, задать словом специального вида, которое называется **мгновенным описанием**, **машинным словом**, **полной конфигурацией** или просто **конфигурацией**.

Мгновенное описание – это слово вида

$$\langle u_1 q_i a_{j_1} v_1, \dots, u_k q_i a_{j_k} v_k \rangle,$$

где q_i – состояние машины Тьюринга в данный момент, a_{j_l} – символ, записанный в обозреваемой ячейке l -ой ленты ($1 \leq l \leq k$), а u_l и v_l – слова, записанные левее и правее этой ячейки, т.е. на этой ленте записано слово $u_l a_{j_l} v_l$.

Мгновенное описание полностью описывает состояние машины Тьюринга в данный момент. При этом будем говорить, что *машина T находится в конфигурации* $\langle u_1 q_i a_{j_1} v_1, \dots, u_k q_i a_{j_k} v_k \rangle$.

Совершенно аналогично тому, как это было сделано в случае одноленточных машин Тьюринга, определяется конфигурация K'_T , в которую *машина перейдет из конфигурации* K_T в результате выполнения соответствующей команды $P_T(i, j_1, \dots, j_k)$ из программы P_T .

Запись $K_T \vdash_T L_T$ будет служить сокращением для утверждения "машина Тьюринга T из конфигурации K_T переходит в конфигурацию L_T за один шаг (за один такт работы, в результате выполнения одной команды)".

Через \vdash_T^* будем обозначать рефлексивное, транзитивное замыкание отношения \vdash_T , т.е. запись $K_T \vdash_T^* L_T$ будет служить сокращением для утверждения "машина Тьюринга T из конфигурации K_T переходит в конфигурацию L_T за конечное число шагов (за конечное число тактов работы)".

Как и в случае одноленточных машин Тьюринга, будем считать состояние q_1 **начальным** состоянием машины Тьюринга.

Чтобы с k -ленточной машиной Тьюринга T связать алгоритм \mathcal{A}_T , преобразующий (перерабатывающий) слова в ее внешнем алфавите \mathcal{A}_T в слова в том же алфавите, необходимо определить, как размещаются исходные данные (аргументы) и как считывается результат. Например, можно считать, что исходные данные размещаются на первой ленте, а результат считывается с k -ой ленты. При этом можно предусмотреть, чтобы содержание первой ленты не менялось, информация с нее лишь считывалась. В этом случае первую ленту можно выделить и назвать *входной лентой*. Однако для наших ближайших целей это не потребуется, так как мы рассмотрим лишь вопросы, связанные с распознаванием языков.

Пусть L – произвольный язык в алфавите Σ , т.е. $L \subseteq \Sigma^*$.

k -ленточная машина Тьюринга

$$T = \langle \mathcal{A}_T, Q_T, P_T, q_0, q_1 \rangle$$

распознает язык L в алфавите Σ , если

- 1) $\Sigma \subseteq \mathcal{A}_T \setminus \{a_0\}$,
- 2) для любого слова w в алфавите Σ :
 - а) если $w \in L$, то, начав работать в конфигурации

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

машина Тьюринга T через конечное число шагов перейдет в заключительную конфигурацию вида

$$\langle q_0 a_0 a_1 a_0 \dots a_0, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle,$$

т.е.

$$\langle q_1 a_0 w a_0, q_1 a_0, \dots, q_1 a_0 \rangle \vdash_T^* \langle q_0 a_0 a_1 a_0 \dots a_0, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle,$$

- б) если $w \notin L$, то, начав работать в конфигурации

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

машина Тьюринга T через конечное число шагов перейдет в заключительную конфигурацию вида

$$\langle q_0 a_0 a_0 a_0 \dots a_0, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle,$$

т.е.

$$\langle q_1 a_0 w a_0, q_1 a_0, \dots, q_1 a_0 \rangle \vdash_T^* \langle q_0 a_0 a_0 a_0 \dots a_0, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle.$$

Нетрудно понять, что множество всех языков над конечным алфавитом Σ , распознаваемых подходящими детерминированными машинами Тьюринга, замкнуто относительно теоретико-множественных операций объединения \cup , пересечения \cap и взятия дополнения $-$. Языки, распознаваемые машинами Тьюринга, имеют разрешимые проблемы вхождения и в силу тезиса Черча – Тьюринга – это в точности рекурсивные языки.

Введем более общее понятие – понятие языка, **допускаемого** (**принимаемого**) k -ленточной машиной Тьюринга

$$T = \langle \mathcal{A}_T, Q_T, P_T, q_0, q_1 \rangle.$$

Пусть L – произвольный язык в алфавите Σ , т.е. $L \subseteq \Sigma^*$.
 k -ленточная машина Тьюринга

$$T = \langle \mathcal{A}_T, Q_T, P_T, q_0, q_1 \rangle$$

допускает (**принимает**) язык L в алфавите Σ , если

- 1) $\Sigma \subseteq \mathcal{A}_T \setminus \{a_0\}$,
- 2) для любого слова w в алфавите Σ :
 - а) если $w \in L$, то, начав работать в конфигурации

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

машина Тьюринга T через конечное число шагов остановится, т.е. перейдет в заключительную конфигурацию вида

$$\langle u_1 q_0 a_{j_1} v_1, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle,$$

т.е.

$$\langle q_1 a_0 w a_0, q_1 a_0, \dots, q_1 a_0 \rangle \vdash_T^* \langle u_1 q_0 a_{j_1} v_1, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle,$$

- б) если $w \notin L$, то, начав работать в конфигурации

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

машина Тьюринга T никогда не остановится.

Класс языков в алфавите Σ , **допускаемых** (**принимаемых**) различными машинами Тьюринга, – это класс всех рекурсивно перечислимых языков в этом алфавите.

Важными характеристиками распознавания языка L в алфавите Σ машиной Тьюринга T являются две функции – *временная сложность* $t_{T,\Sigma}(n)$ и *емкостная сложность* $s_{T,\Sigma}(n)$.

Временная сложность $t_{T,\Sigma}(n)$ распознавания языка L в алфавите Σ k -ленточной машиной Тьюринга T – это максимальное число тактов ее работы на всевозможных начальных конфигурациях вида

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

где w – произвольное слово в алфавите Σ длины не более n , до прихода в заключительное состояние q_0 .

Емкостная сложность $s_{T,\Sigma}(n)$ распознавания языка L в алфавите Σ k -ленточной машины Тьюринга T – это максимальное число ячеек на ее лентах, которые были рассмотрены соответствующими головками в ходе ее работы на всевозможных начальных конфигурациях вида

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

где w – произвольное слово в алфавите Σ длины не более n , до прихода в заключительное состояние q_0 .

Легко понять, что для любого n выполняется неравенство

$$s_{T,\Sigma}(n) \leq t_{T,\Sigma}(n).$$

Следующая теорема устанавливает связь между распознаваемостью языков многоленточными и одноленточными детерминированными машинами Тьюринга.

Теорема 3.1. *Если язык L в алфавите Σ распознается некоторой k -ленточной детерминированной машиной Тьюринга T с временной сложностью $t_{T,\Sigma}(n)$, то он распознается подходящей одноленточной детерминированной машиной Тьюринга T_1 с временной сложностью $O(t_{T,\Sigma}^2(n))$.*

Доказательство теоремы основано на моделировании k -ленточной машины Тьюринга T одноленточной с $2k$ "дорожками", причем k "дорожек" содержат записи, имеющиеся на k лентах моделируемой машины Тьюринга, а остальные k "дорожек" служат для указания положения головок на соответствующих лентах исходной машины T . С деталями доказательства можно ознакомиться, например, по [4].

Особую роль играют языки, распознаваемые детерминированными машинами Тьюринга за полиномиальное время. Класс всех таких языков обозначается через $\mathcal{P} - \text{TIME}$.

Язык L в алфавите Σ принадлежит классу языков $\mathcal{P} - \text{TIME}$ тогда и только тогда, когда существует такая k -ленточная машина Тьюринга T и такой полином $p(n)$ с натуральными коэффициентами, что T **распознает** язык L в алфавите Σ и для любого натурального числа n выполняется неравенство $t_{T,\Sigma}(n) \leq p(n)$.

Нетрудно понять, что множество всех языков над конечным алфавитом Σ , распознаваемых детерминированными машинами Тьюринга за полиномиальное время, замкнуто относительно теоретико-множественных операций объединения \cup , пересечения \cap и взятия дополнения $-$.

Аналогично определяется класс $\mathcal{P} - SPACE$ языков, распознаваемых детерминированными машинами Тьюринга с полиномиальной емкостной сложностью.

Так как в дальнейшем не предполагается рассматривать класс $\mathcal{P} - SPACE$, то класс $\mathcal{P} - TIME$ будем обозначать просто как \mathcal{P} .

Можно было бы привести много интересных и содержательных примеров языков из класса \mathcal{P} . Например, для любой конечной сигнатуры τ классу \mathcal{P} принадлежат языки, состоящие из всех термов и всех формул языка первого порядка L_τ . Однако аналогичное нельзя утверждать о языке, состоящем из всех формул, выводимых в Исчислении Предикатов сигнатуры τ .

Непосредственным следствием теоремы 3.1 является следующая теорема

Теорема 3.2. *Если язык L в алфавите Σ распознается некоторой k -ленточной детерминированной машиной Тьюринга T за полиномиальное время, то он распознается за полиномиальное время подходящей одноленточной детерминированной машиной Тьюринга T_1 .*

Рассмотрим достаточно простой и естественный пример из пособия [54] разрешимого языка, не принадлежащего классу \mathcal{P} .

Ранее с каждой детерминированной одноленточной машиной Тьюринга T достаточно естественным образом было связано натуральное число $n(T)$ – **номер** машины Тьюринга T . Из определения $n(T)$ легко получить алгоритм, позволяющий по произвольному натуральному числу n определить, является ли оно номером некоторой машины Тьюринга T , и в случае положительного ответа восстановить саму машину Тьюринга T , т.е. ее внешний \mathcal{A}_T и внутренний \mathcal{Q}_T алфавиты и программу P_T .

По сложившейся традиции будем обозначать a_0 через 0, а a_1 – через 1.

Рассмотрим следующее множество натуральных чисел

$$E = \{ n(T) \mid \text{из конфигурации } q_1 0 1^{n(T)} 0, \\ \text{сделав не более чем } 2^{n(T)} \text{ шагов, машина Тьюринга } T \\ \text{перейдет в заключительную конфигурацию вида } q_0 0 1 0 \dots 0 \}.$$

Предположим, что множество E принадлежит классу \mathcal{P} , т.е. проблема вхождения в него разрешима некоторой детерминированной машиной Тьюринга M в полиномиальное время.

Тогда и дополнение $\overline{E} = N \setminus E$ множества E принадлежит классу \mathcal{P} , т.е. проблема вхождения в него также разрешима некоторой детерминированной машиной Тьюринга T в полиномиальное время.

В качестве последней можно взять композицию машины Тьюринга M и следующей машины M_1 , выполняющей преобразования

$$\begin{array}{l} q_1 010 \dots 0 \mid \xRightarrow{T} q_0 000 \dots 0, \\ q_1 000 \dots 0 \mid \xRightarrow{T} q_0 010 \dots 0. \end{array}$$

$$\begin{array}{ll} q_1 0 \longrightarrow q_2 0R, & q_2 1 \longrightarrow q_0 0L, \\ q_2 0 \longrightarrow q_0 1L. \end{array}$$

Пусть полином $p(n)$ с натуральными коэффициентами ограничивает время распознавания языка \overline{E} детерминированной машиной Тьюринга T . Тогда для любого натурального числа m машина Тьюринга T из конфигурации $q_1 01^m 0$ не более чем за $p(m)$ шагов перейдет в заключительную конфигурацию вида $q_0 0\varepsilon 0 \dots 0$, где $\varepsilon = 1$, если $n \in \overline{E}$, т.е. $n \notin E$, и $\varepsilon = 0$, если $n \notin \overline{E}$, т.е. $n \in E$.

Существует такое число s , что при $n > s$ выполняется неравенство $p(s) < 2^s$.

Вводя, в случае необходимости, *новые внутренние состояния* q_i и добавляя к программе машины Тьюринга T новые команды вида

$$q_i a_j \rightarrow q_i a_j S,$$

мы получим машину T_1 , которая на конфигурациях вида $q_1 01^m 0$ работает как машина Тьюринга T и при этом $n(T_1) > s$. Чтобы не усложнять обозначения, будем машину Тьюринга T_1 обозначать через T . Поэтому считаем, что выполнены неравенства $n(T) > s$ и $p(n(T)) < 2^{n(T)}$.

Полагаем $n_0 = n(T)$.

Предположим, что $n_0 \in \overline{E}$ (т.е. $n_0 \notin E$). Тогда машина Тьюринга T из конфигурации $q_1 01^{n_0} 0$, т.е. из конфигурации $q_1 01^{n(T)} 0$, не более чем за $p(n_0) = p(n(T))$ шагов перейдет в заключительную конфигурацию вида $q_0 010 \dots 0$. Так как $p(n(T)) < 2^{n(T)}$, то $n_0 = n(T) \in E$. Полученное противоречие означает, что $n_0 \notin \overline{E}$ (т.е. $n_0 \in E$).

Тогда, с одной стороны, машина Тьюринга T из конфигурации $q_1 01^{n_0} 0$, т.е. из конфигурации $q_1 01^{n(T)} 0$, не более чем за $p(n_0) = p(n(T)) < 2^{n(T)}$ шагов перейдет в заключительную конфигурацию вида $q_0 000 \dots 0$. А с другой стороны, машина Тьюринга T из конфигурации $q_1 01^{n_0} 0$, т.е. из конфигурации $q_1 01^{n(T)} 0$, не более чем за $2^{n(T)}$ шагов перейдет в заключительную конфигурацию вида $q_0 010 \dots 0$.

Полученное противоречие показывает, что *множество E не принадлежит классу \mathcal{P} , т.е. проблема вхождения в него не разрешима никакой детерминированной машиной Тьюринга M в полиномиальное время.*

В конце XX века особое внимание исследователей было приковано к изучению класса \mathcal{NP} языков, определение которого аналогично определению класса \mathcal{P} , но основано на некотором обобщении рассмотренного выше понятия машины Тьюринга – на понятии *k -ленточной недетерминированной* машины Тьюринга, к введению которого мы и переходим.

k -ленточная *недетерминированная машина Тьюринга* T , как и детерминированная, имеет *внешний алфавит*, или *алфавит ленточных символов*,

$$\mathcal{A}_T = \{a_0, a_1, \dots, a_n\},$$

внутренний алфавит, или *алфавит внутренних состояний*,

$$Q_T = \{q_0, q_1, \dots, q_m\},$$

программу P_T .

Элементы внешнего алфавита называются просто *символами*, а элементы внутреннего алфавита – *внутренними состояниями*.

У k -ленточной недетерминированной машины Тьюринга, как и у детерминированной, имеется k *лент*, каждая из которых разбита на конечное число ячеек, в которые можно записывать символы из внешнего алфавита $\{a_0, a_1, \dots, a_n\}$. Среди ленточных символов имеется так называемый *пустой символ*, который традиционно обозначают через a_0 . Ячейки ленты упорядочены слева направо. Мы рассматриваем вариант определения машины Тьюринга, в котором предполагается, что ленты *потенциально бесконечны в обе стороны*. Это означает, что в процессе работы машины Тьюринга разрешается добавлять новые пустые ячейки как к правому, так и к левому концам любой ленты (считается, что в момент добавления в них записан пустой символ a_0).

С каждой лентой связана своя *головка*, которая в каждый момент времени обозревает одну ячейку соответствующей ленты, может считывать обозреваемый символ и записывать в обозреваемую ячейку новый символ.

У недетерминированной машины Тьюринга имеется управляющее устройство, которое в каждый момент времени находится в одном из *внутренних состояний* q_0, q_1, \dots, q_m . Состояние q_0 традиционно называется *заключительным состоянием*.

Недетерминированная машина Тьюринга, как и детерминированная, выполняет действия потактово, т.е. в дискретные моменты времени.

Работа машины Тьюринга состоит из отдельных тактов, выполняемых согласно ее программе P_T .

Программа P_T для k -ленточной недетерминированной машины Тьюринга T – это отображение δ_T множества $(Q_T \setminus \{q_0\}) \times \mathcal{A}_T^k$ во множество

$$P(Q_T \times (\mathcal{A}_T \times \{L, S, R\})^k)$$

всех подмножеств множества $Q_T \times (\mathcal{A}_T \times \{L, S, R\})^k$

$$\delta_T : (Q_T \setminus \{q_0\}) \times \mathcal{A}_T^k \longrightarrow (Q_T \times (\mathcal{A}_T \times \{L, S, R\})^k).$$

Как и в детерминированном случае, можно считать, что программа P_T недетерминированной машины Тьюринга T – это конечный набор команд вида:

$$q_i \langle a_{j_1}, \dots, a_{j_k} \rangle \longrightarrow q_s \langle \langle a_{t_1}, D_1 \rangle, \dots, \langle a_{t_k}, D_k \rangle \rangle,$$

где $D_1, \dots, D_k \in \{L, S, R\}$.

При этом произвольному набору $q_i \langle a_{j_1}, \dots, a_{j_k} \rangle$ при $i > 0$ в отличие от детерминированного случая может соответствовать несколько команд, в которых перед \longrightarrow стоит этот набор. Весь этот набор команд мы будем обозначать через $P_T(i, j_1, \dots, j_k)$.

Если в текущий момент времени недетерминированная машина Тьюринга находится в состоянии q_i при $i > 0$, а в обозреваемых ячейках ленты находятся символы a_{j_1}, \dots, a_{j_k} , то машина **может выполнить** любую команду из набора $P_T(i, j_1, \dots, j_k)$.

Напомним, что в случае детерминированной машины Тьюринга в ситуации, аналогичной рассматриваемой, машина **должна выполнить** единственную команду $P_T(i, j_1, \dots, j_k)$, т.е. ту единственную команду, в которой перед \longrightarrow стоит набор $q_i \langle a_{j_1}, \dots, a_{j_k} \rangle$.

При этом, если выбранная машиной для исполнения команда имеет вид

$$q_i \langle a_{j_1}, \dots, a_{j_k} \rangle \longrightarrow q_s \langle \langle a_{t_1}, D_1 \rangle, \dots, \langle a_{t_k}, D_k \rangle \rangle,$$

то

- 1) машина переходит в состояние q_s ,
- 2) на l -ой ($1 \leq l \leq k$) ленте выполняется команда

$$q_i a_{j_l} \longrightarrow q_s a_{t_l} D_l,$$

т.е. в обозреваемой ячейке символ a_{j_l} заменяется на символ a_{t_l} и выполняется действие D_l как для одноленточной машины Тьюринга.

Если в результате выполнения некоторой команды машина *приходит в заключительное состояние* q_0 , то говорят, что она **останавливается**, так как по определению ни одна команда программы не начинается с q_0 .

В некоторых ситуациях удобно программу P_T рассматривать как *отношение* Δ_T на множестве

$$(Q_T \setminus \{q_0\}) \times A_T^k \times Q_T \times (\mathcal{A}_T \times \{L, S, R\})^k,$$

т.е.

$$\Delta_T \subseteq (Q_T \setminus \{q_0\}) \times A_T^k \times Q_T \times (\mathcal{A}_T \times \{L, S, R\})^k.$$

Как и в детерминированном случае, полное описание состояния k -ленточной недетерминированной машины Тьюринга в данный момент времени можно задать словом специального вида, которое называется **мгновенным описанием, машинным словом, полной конфигурацией** или просто **конфигурацией**.

Мгновенное описание — это слово вида

$$\langle u_1 q_i a_{j_1} v_1, \dots, u_k q_i a_{j_k} v_k \rangle,$$

где q_i — состояние машины Тьюринга в данный момент, a_{j_l} — символ, записанный в обозреваемой ячейке l -ой ленты ($1 \leq l \leq k$), а u_l и v_l — слова, записанные левее и правее этой ячейки, т.е. на этой ленте записано слово $u_l a_{j_l} v_l$.

Мгновенное описание полностью описывает состояние машины Тьюринга в данный момент. При этом будем говорить, что *машина T находится в конфигурации $\langle u_1 q_i a_{j_1} v_1, \dots, u_k q_i a_{j_k} v_k \rangle$* .

Подобно тому, как это было сделано в случае детерминированных машин Тьюринга, определяется конфигурация K'_T , в которую машина **может перейти** из конфигурации K_T в результате выполнения одной команды из $P_T(i, j_1, \dots, j_k)$ или одной из команд программы P_T .

Запись $K_T \vdash_T L_T$ будет служить сокращением для утверждения "машина Тьюринга T из конфигурации K_T **может перейти** в конфигурацию L_T за один шаг (за один такт работы, в результате выполнения одной команды)".

Через \vdash_T^* будем обозначать рефлексивное, транзитивное замыкание отношения \vdash_T , т.е. запись $K_T \vdash_T^* L_T$ будет служить сокращением для утверждения "машина Тьюринга T из конфигурации K_T **может перейти** в конфигурацию L_T за конечное число шагов (за конечное число тактов работы)".

Как и в случае детерминированных машин Тьюринга, будем считать состояние q_1 **начальным** состоянием.

Как и в детерминированном случае, рассмотрим вопрос о **распознавании** языков недетерминированными машинами Тьюринга.

Пусть L – произвольный язык в алфавите Σ , т.е. $L \subseteq \Sigma^*$.

k -ленточная недетерминированная машина Тьюринга

$$T = \langle \mathcal{A}_T, Q_T, P_T, q_0, q_1 \rangle$$

распознает язык L в алфавите Σ , если

- 1) $\Sigma \subseteq \mathcal{A}_T \setminus \{a_0\}$,
- 2) для любого слова w в алфавите Σ :
 - а) *любое вычисление*, начинающееся в конфигурации

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

завершается недетерминированной машиной Тьюринга T через конечное число шагов, т.е. в этом случае любой путь вычислений ведет в заключительную конфигурацию вида

$$\langle u_1 q_0 a_{j_1} v_1, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle,$$

- б1) если $w \in L$, то, начав работать в конфигурации

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

недетерминированная машина Тьюринга T через конечное число шагов **может перейти** в заключительную конфигурацию вида

$$\langle q_0 a_0 a_1 a_0 \dots a_0, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle,$$

- б2) если $w \notin L$, то, начав работать в конфигурации

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

недетерминированная машина Тьюринга T *не может перейти* через конечное число шагов в заключительную конфигурацию вида

$$\langle q_0 a_0 a_1 a_0 \dots a_0, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle.$$

Нетрудно понять, что множество всех языков над конечным алфавитом Σ , распознаваемых подходящими недетерминированными машинами Тьюринга, замкнуто относительно теоретико множественных операций объединения \cup , пересечения \cap и взятия дополнения $-$. Языки, распознаваемые недетерминированными машинами Тьюринга, имеют разрешимые проблемы вхождения и в силу тезиса Черча – Тьюринга – это в точности рекурсивные языки.

Как и в детерминированном случае важными характеристиками распознавания языка L в алфавите Σ недетерминированной машиной Тьюринга T являются две функции – *временная сложность* $t_{T,\Sigma}(n)$ и *емкостная сложность* $s_{T,\Sigma}(n)$. Однако при их определении требуется некоторая аккуратность. Так как нашей ближайшей целью будет обсуждение вопроса о сложности распознавания языков (о сложности решения проблемы вхождения для языков), то можно дать следующие определения.

Временная сложность $t_{T,\Sigma}(n)$ распознавания языка L в алфавите Σ k -ленточной недетерминированной машиной Тьюринга T – это максимальное число тактов ее работы на всевозможных начальных конфигурациях вида

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

где w – произвольное слово в алфавите Σ длины не более n , до прихода в заключительное состояние q_0 .

Следующая теорема устанавливает связь между распознаваемостью языков многоленточными и одноленточными недетерминированными машинами Тьюринга.

Теорема 3.3. *Если язык L в алфавите Σ распознается некоторой k -ленточной недетерминированной машиной Тьюринга T с временной сложностью $t_{T,\Sigma}(n)$, то он распознается подходящей одноленточной недетерминированной машиной Тьюринга T_1 с временной сложностью $O(t_{T,\Sigma}^2(n))$.*

Доказательство теоремы основано на моделировании k -ленточной недетерминированной машины Тьюринга T одноленточной с $2k$ "дорожками", причем k "дорожек" содержат записи, имеющиеся на k лентах моделируемой машины Тьюринга, а остальные k "дорожек" служат для указания положения головок на соответствующих лентах исходной машины T . С деталями доказательства можно ознакомиться, например, по [4].

Существует определенная связь между распознаваемостью языков недетерминированными и детерминированными машинами Тьюринга. Мы не предполагаем детально рассмотреть этот вопрос, а лишь ограничимся формулировкой теоремы.

Функция $f(n)$ называется *конструируемой по времени*, если существует такая детерминированная машина Тьюринга T , которая на любом входе длины n делает ровно $f(n)$ шагов до остановки.

Теорема 3.4. *Если язык L в алфавите Σ распознается некоторой недетерминированной машиной Тьюринга T с временной сложностью $t_{T,\Sigma}(n)$ и функция $t_{T,\Sigma}(n)$ конструируема по времени, то он распознается подходящей детерминированной машиной Тьюринга T_1 с временной сложностью $c^{t_{T,\Sigma}(n)}$, где c – некоторая константа.*

Во второй половине XX века интенсивно изучались языки, распознаваемые недетерминированными машинами Тьюринга за полиномиальное время. Класс всех таких языков обозначается через $\mathcal{NP} - TIME$.

Язык L в алфавите Σ принадлежит классу языков $\mathcal{NP} - TIME$ тогда и только тогда, когда существует такая k -ленточная недетерминированная машина Тьюринга T и такой полином $p(n)$ с натуральными коэффициентами, что T **распознает** язык L в алфавите Σ и для любого натурального числа n выполняется неравенство $t_{T,\Sigma}(n) \leq p(n)$.

Ввиду особой важности класса языков $\mathcal{NP} - TIME$ сделаем некоторые пояснения.

Пусть L – произвольный язык в алфавите Σ , т.е. $L \subseteq \Sigma^*$. Он принадлежит классу языков $\mathcal{NP} - TIME$ тогда и только тогда, когда существует такая k -ленточная недетерминированная машина Тьюринга

$$T = \langle \mathcal{A}_T, Q_T, P_T, q_0, q_1 \rangle$$

и такой полином $p(n)$ с натуральными коэффициентами, что

- 1) $\Sigma \subseteq \mathcal{A}_T \setminus \{a_0\}$,
- 2) для любого слова w в алфавите Σ :
 - а) *любое вычисление*, начинающееся в конфигурации

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

завершается недетерминированной машиной Тьюринга T через не более чем $p(|w|)$ шагов, т.е. в этом случае любой путь вычислений ведет через не более чем $p(|w|)$ шагов в заключительную конфигурацию вида

$$\langle u_1 q_0 a_{j_1} v_1, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle,$$

- б1) если $w \in L$, то, начав работать в конфигурации

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

недетерминированная машина Тьюринга T через не более чем $p(|w|)$ шагов **может перейти** в заключительную конфигурацию вида

$$\langle q_0 a_0 a_1 a_0 \dots a_0, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle,$$

т.е. в этом случае по крайней мере одна из заключительных конфигураций имеет указанный вид,

б2) если $w \notin L$, то, начав работать в конфигурации

$$\langle q_0 a_0 w a_0, q_0 a_0, \dots, q_0 a_0 \rangle,$$

недетерминированная машина Тьюринга T **не может перейти** через конечное число шагов в заключительную конфигурацию вида

$$\langle q_0 a_0 a_1 a_0 \dots a_0, u_2 q_0 a_{j_2} v_2, \dots, u_k q_0 a_{j_k} v_k \rangle,$$

т.е. в этом случае ни одна из заключительных конфигураций не имеет указанный вид.

Аналогично определяется класс $\mathcal{NP} - SPACE$ языков, распознаваемых недетерминированными машинами Тьюринга с полиномиальной емкостной сложностью.

Так как в дальнейшем не предполагается рассматривать класс $\mathcal{NP} - SPACE$, то класс $\mathcal{NP} - TIME$ будем обозначать просто как \mathcal{NP} .

Непосредственным следствием теоремы 3.3 является следующая теорема.

Теорема 3.5. *Если язык L в алфавите Σ распознается некоторой k -ленточной недетерминированной машиной Тьюринга T за полиномиальное время, то он распознается за полиномиальное время подходящей одноленточной недетерминированной машиной Тьюринга T_1 .*

Существует много интересных и содержательных примеров языков из класса \mathcal{NP} . Для знакомства с ними можно порекомендовать замечательную книгу М. Гэри и Д. Джонсона [15].

Так как каждая детерминированная машина Тьюринга может рассматриваться как частный случай недетерминированной, то

$$\mathcal{P} \subseteq \mathcal{NP}.$$

Вопрос о том, выполняется ли равенство $\mathcal{NP} = \mathcal{P}$, стал во второй половине XX века одной из самых знаменитых проблем математики и информатики. Именно этот вопрос открывает известный список из семи проблем **"Проблемы третьего тысячелетия"**.

При изучении класса языков \mathcal{NP} , в частности, с целью получения ответа на вопрос, выполняется ли равенство $\mathcal{NP} = \mathcal{P}$, было введено оказавшееся в дальнейшем очень важным и полезным, понятие **полиномиальной сводимости** одного языка к другому. Полиномиальная сводимость языков – это весьма частный случай понятия **m -сводимость**, хорошо нам известного из теории рекурсивно перечислимых множеств. Было бы более точно говорить о **сводимости проблемы вхождения в один язык к проблеме вхождения в другой язык**, однако мы будем придерживаться сложившейся терминологии.

Язык L в алфавите Σ_L **полиномиально сводим** к языку S в алфавите Σ_S , если существует одноленточная детерминированная машина Тьюринга T с внешним алфавитом $\mathcal{A}_T \supseteq \Sigma_L \cup \Sigma_S$ и полином $p(n)$, такие, что для любого слова w в алфавите Σ_L машина T , начав работать в конфигурации $q_1 a_0 w a_0$, остановится в конфигурации вида $q_1 a_0 u a_0 \dots a_0$, где $u \in \Sigma_S^*$, сделав не более чем $p(|w|)$ шагов, и имеет место эквивалентность

$$w \in L \iff u \in S.$$

Заметим, что при этом выполняется неравенство $|u| \leq p(|w|)$.

Изучение класса \mathcal{NP} привело, в частности, к введению оказавшегося чрезвычайно важным и полезным понятия **\mathcal{NP} -полного языка**.

Язык L из класса \mathcal{NP} называется **\mathcal{NP} -полным**, если к нему полиномиально сводим любой язык S из этого класса \mathcal{NP} .

Обширный список **\mathcal{NP} -полных** языков читатель найдет в уже цитировавшейся выше замечательной книге М. Гэри и Д. Джонсона [15]. Ряд важных **\mathcal{NP} -полных** языков, относящихся к различным разделам математики, будет рассмотрен ниже.

Заметим, что обозначение \mathcal{NP} служит сокращением для "недетерминированный полиномиальный" (*nondeterministic polynomial*), что связано с первоначальным определением этого класса языков через вычисления на недетерминированных машинах Тьюринга. Напомним, что понятие недетерминированного конечного автомата оказалось весьма полезным при изучении класса автоматных языков, в частности, при доказательстве теоремы С. Клини о совпадении классов автоматных и регулярных языков.

В то же время существует в определенном смысле более удобный подход к определению класса языков \mathcal{NP} .

Язык L в алфавите Σ принадлежит классу языков $\mathcal{NP} - \text{TIME}$ тогда и только тогда, когда существует такая детерминированная машина Тьюринга T и такие полиномы $q(n)$ и $p(n)$ с натуральными коэффициентами, что

1) $\Sigma \subseteq \mathcal{A}_T \setminus \{a_0\}$,

2) для любых слов w в алфавите Σ и u в алфавите $\mathcal{A}_T \setminus \{a_0\}$ детерминированная машина Тьюринга T , начав работать в конфигурации $q_1 a_0 w a_0 u a_0$, остановится, сделав не более чем $p(|w| + |u| + 1)$ шагов,

3) для любого слова w в алфавите Σ справедлива эквивалентность:

слово w принадлежит языку L тогда и только тогда, когда **существует** такое слово v в алфавите $\mathcal{A}_T \setminus \{a_0\}$, что $|v| \leq q(|w|)$ и, начав работать в конфигурации $q_1 a_0 w a_0 v a_0$, детерминированная машина Тьюринга T остановится в конфигурации $q_0 a_0 a_1 a_0 \dots a_0$ (сделав при этом не более чем $p(|w| + |v| + 1)$ шагов).

Сделаем некоторые пояснения.

Слово v , о котором идет речь в пункте 3) называют **сертификатом** или **удостоверением**.

Пункт 3) утверждает, что если слово w принадлежит языку L , то существует "короткий" сертификат v ($|v| \leq q(|w|)$), подтверждающий это.

Пункт 2) утверждает, что машина Тьюринга T "быстро проверяет" сертификаты.

Таким образом,

если слово w принадлежит языку L , то найдется такой "короткий" сертификат v , что за не более чем $p(|w| + |v| + 1)$ тактов работы машина Тьюринга T из начальной конфигурации $q_1 a_0 w a_0$ перейдет в заключительную конфигурацию $q_0 a_0 a_1 a_0 \dots a_0$.

Так как $|v| \leq q(|w|)$, то

если слово w принадлежит языку L , то за не более чем $p(|w| + q(|w|) + 1)$ тактов работы машина Тьюринга T из начальной конфигурации $q_1 a_0 w a_0$ перейдет в заключительную конфигурацию $q_0 a_0 a_1 a_0 \dots a_0$.

Заметим, что для любых двух полиномов $p(n)$ и $q(n)$ с натуральными коэффициентами $p(n + q(n) + 1)$ – полином с натуральными коэффициентами.

Поясним, почему эти два определения задают один и тот же класс языков.

Предположим, что язык L в алфавите Σ принадлежит классу языков $\mathcal{NP} - \text{TIME}$ в соответствии со вторым определением этого класса. Тогда существует такая машина Тьюринга T и такие полиномы $q(n)$ и $p(n)$ с натуральными коэффициентами, что

1) $\Sigma \subseteq \mathcal{A}_T \setminus \{a_0\}$,

2) для любых слов w в алфавите Σ и v в алфавите $\mathcal{A}_T \setminus \{a_0\}$ детерминированная машина Тьюринга T , начав работать в конфигурации $q_1 a_0 w a_0 v a_0$, остановится, сделав не более чем $p(|w| + |v| + 1)$ шагов,

3) для любого слова w в алфавите Σ справедлива эквивалентность:

слово w принадлежит языку L тогда и только тогда, когда **существует** такое слово v в алфавите $\mathcal{A}_T \setminus \{a_0\}$, что $|v| \leq q(|w|)$ и, начав работать в конфигурации $q_1 a_0 w a_0 v a_0$, детерминированная машина Тьюринга T остановится в конфигурации $q_0 a_0 a_1 a_0 \dots a_0$ (сделав при этом не более чем $p(|w| + |v| + 1)$ шагов).

Нетрудно построить недетерминированную машину Тьюринга T_0 с внешним алфавитом \mathcal{A}_T , которая из конфигурации $q_1 a_0 w a_0$ может за конечное число шагов перейти в любую конфигурацию вида $q_0 a_0 w a_0 v a_0$, где v – любое слово в алфавите $\mathcal{A}_T \setminus \{a_0\}$.

Тогда суперпозиция $T_0 T$ машин Тьюринга T_0 и T является недетерминированной машиной Тьюринга с полиномом $2(n + 1 + q(n)) + p(n + q(n) + 1)$, ограничивающим время ее работы, и распознающей исходный язык L (некоторые детали оставлены читателю в качестве полезного упражнения).

Обратно, пусть для языка L в алфавите Σ существует такая одноленточная недетерминированная машина Тьюринга

$$T = \langle \mathcal{A}_T, Q_T, P_T, q_0, q_1 \rangle$$

и такой полином $p(n)$ с натуральными коэффициентами, что

1) $\Sigma \subseteq \mathcal{A}_T \setminus \{a_0\}$,

2) для любого слова w в алфавите Σ :

а) *любое вычисление*, начинающееся в конфигурации $q_0 a_0 w a_0$, завершается недетерминированной машиной Тьюринга T через не более чем $p(|w|)$ шагов, т.е. в этом случае любой путь вычислений ведет через не более чем $p(|w|)$ шагов в заключительную конфигурацию вида $u_1 q_0 a_{j_1} v_1$,

б1) если $w \in L$, то, начав работать в конфигурации $q_0 a_0 w a_0$, недетерминированная машина Тьюринга T через не более чем $p(|w|)$ шагов **может перейти** в заключительную конфигурацию вида $q_0 a_0 a_1 a_0 \dots a_0$, т.е. в этом случае по крайней мере одна из заключительных конфигураций имеет указанный вид,

б2) если $w \notin L$, то, начав работать в конфигурации $q_0 a_0 w a_0$, недетерминированная машина Тьюринга T **не может перейти** через конечное число шагов в заключительную конфигурацию вида $q_0 a_0 a_1 a_0 \dots a_0$, т.е. в этом случае ни одна из заключительных конфигураций не имеет указанный вид.

Выберем новый символ $*$, не входящий в алфавит \mathcal{A}_T , для разделения слов в этом алфавите.

На множестве всех слов в алфавите $\mathcal{A}_T \cup \{*\}$ определим двуместный предикат $T(x, y)$ следующей эквивалентностью:

для слов w и v в алфавите $\mathcal{A}_T \cup \{*\}$ выполнено $T(w, v)$ тогда и только тогда, когда

1) w – слово в алфавите Σ ,

2) слово v представимо в виде $v_0 * v_1 * \dots * v_s$, где каждое слово v_t ($0 \leq t \leq s$) является мгновенным описанием (конфигурацией) для машины T , причем v_0 – это начальная конфигурация $q_1 a_0 w a_0$, а v_s – это заключительная конфигурация $q_0 a_0 a_1 a_0 \dots a_0$ и при любом t ($0 \leq t \leq s-1$) $v_t \vdash_T v_{t+1}$.

Тогда слово w принадлежит языку, распознаваемому недетерминированной машиной Тьюринга T , тогда и только тогда, когда существует "короткий" сертификат v указанного вида

(так как $s \leq p(|w|)$ и при любом t ($0 \leq t \leq s$) $|v_t| \leq |w| + 1 + s \leq |w| + 1 + p(|w|)$), то

$$|v| \leq s + (s+1)(|w| + 1 + s) \leq p(|w|) + (p(|w|) + 1)(|w| + 1 + p(|w|)) \leq q(|w|),$$

где $q(n)$ – подходящий полином).

Нетрудно понять, что время "проверки сертификата" v ограничено сверху подходящим полиномом.

ГЛАВА V

\mathcal{NP} -ТРУДНЫЕ И \mathcal{NP} -ПОЛНЫЕ ПРОБЛЕМЫ

§1. \mathcal{NP} -полнота проблемы выполнимости для формул логики высказываний

Этот параграф посвящен фундаментальной теореме С. Кука об \mathcal{NP} -полноте проблемы выполнимости для формул логики высказываний. В изложении доказательства мы придерживаемся подхода из монографии [4].

Теорема 1.1 (С. Кук). *Проблема выполнимости для формул логики высказываний является \mathcal{NP} -полной.*

Д о к а з а т е л ь с т в о. Обозначим через Σ алфавит Логики высказываний

$$\Sigma = V \cup \Sigma_1 \cup \Sigma_2,$$

где $V = \{A_1, A_2, \dots, A_n, \dots\}$ – счетное множество пропозициональных переменных, $\Sigma_1 = \{\neg, \vee, \&\}$ – множество пропозициональных (логических) связок, а множество $\Sigma_2 = \{(,)\}$ состоит из левой и правой скобок.

Через L обозначим множество всех выполнимых формул Логики высказываний. Тогда $L \subseteq \Sigma^*$ и нетрудно понять, что *язык L принадлежит классу \mathcal{NP}* (если Φ – выполнимая формула Логики высказываний, то "коротким сертификатом", подтверждающим принадлежность формулы Φ языку L , может служить любой набор истинностных значений входящих в эту формулу пропозициональных переменных, на котором формула Φ истинна).

Покажем, что *проблема вхождения для произвольного языка из класса \mathcal{NP} полиномиально сводима к проблеме выполнимости для формул логики высказываний, т.е. к проблеме вхождения в язык L .*

Пусть L – произвольный язык, проблема вхождения в который решается за полиномиальное время $p(n)$ некоторой недетерминированной одноленточной машиной Тьюринга T , имеющей внешний алфавит

$$A_T = \{a_0, a_1, \dots, a_m\},$$

внутренний алфавит

$$Q_T = \{q_0, q_1, \dots, q_s\}$$

и функцию переходов

$$\delta : Q_T \setminus \{q_0\} \times A_t \longrightarrow P(Q_T \times A_T \times \{L, S, R\}).$$

По чисто техническим причинам истинностные значения будем обозначать не через **И** и **Л**, а через 1 и 0.

Для произвольного натурального числа p через $U(A_1, \dots, A_p)$ обозначим следующую формулу

$$(A_1 \vee \dots \vee A_p) \& \bigwedge_{1 \leq i < j \leq p} (\neg A_i \vee \neg A_j).$$

Ясно, что

для произвольного набора $\langle \varepsilon_1, \dots, \varepsilon_p \rangle$ истинностных значений равенство $U(\varepsilon_1, \dots, \varepsilon_p) = 1$ выполнено тогда и только тогда, когда найдется единственное число i_0 такое, что $\varepsilon_{i_0} = 1$.

Считаем, что ячейки ленты машины Тьюринга T занумерованы слева направо неотрицательными целыми числами. Для описания работы машины Тьюринга T мы будем использовать пропозициональные переменные. Однако будем их обозначать таким образом, чтобы облегчить понимание смысла проводимых построений – вместо несколько "обезличенного" обозначения A_n мы будем использовать обозначения $H\langle i, t \rangle$, $C\langle i, j, t \rangle$ и $S\langle j, t \rangle$, смысл которых будет ясен позже.

1) Построим формулу \mathcal{A} , содержательно утверждающую, что в каждый момент времени $0 \leq t \leq p(n)$ машина Тьюринга T обозревает в точности одну ячейку ленты. Для построения формулы \mathcal{A} рассмотрим счетное множество пропозициональных переменных

$$\{H\langle i, t \rangle \mid i, t = 0, 1, 2, \dots\}.$$

Содержательно истинность переменной $H\langle i, t \rangle$ означает, что в момент времени t машина Тьюринга T обозревает ячейку с номером i .

Полагаем

$$\mathcal{A}_t \equiv U(H\langle 0, t \rangle, H\langle 1, t \rangle, \dots, H\langle p(n), t \rangle), \quad \mathcal{A} \equiv \mathcal{A}_0 \& \mathcal{A}_1 \& \dots \& \mathcal{A}_{p(n)}.$$

Если мы считаем, что каждая пропозициональная переменная имеет длину 1, то длина формулы \mathcal{A} есть $O(p^3(n))$.

Если же считать, что каждая пропозициональная переменная $H\langle k, t \rangle$ имеет длину $1 + 1 + \log k + 1 + \log t + 1$, т.е. $O(\log p(n))$, то длина формулы \mathcal{A} есть $O(p^4(n))$. В этом случае числа k и t задаются в двоичной системе. Наконец, если числа k и t задать в унарной системе, т.е. как слова соответственно длин k и t в однобуквенном алфавите $\{|\}$, то каждая пропозициональная переменная $H\langle k, t \rangle$ будет иметь длину $1 + 1 + k + 1 + t + 1$, т.е. $O(p(n))$, и вновь длина формулы \mathcal{A} есть $O(p^4(n))$.

Так как нас интересует полиномиальная сводимость, то в дальнейшем для упрощения считаем, что каждая пропозициональная переменная имеет длину 1. В противном случае надо лишь добавлять множитель $p(n)$.

2) Построим формулу \mathcal{B} , содержательно утверждающую, что *в каждый момент времени $0 \leq t \leq p(n)$ каждая ячейка ленты содержит в точности один символ*. Для построения формулы \mathcal{B} рассмотрим счетное множество пропозициональных переменных

$$\{ C\langle i, j, t \rangle \mid i, j, t = 0, 1, 2, \dots \}.$$

Содержательно истинность переменной $C\langle i, j, t \rangle$ означает, что *в момент времени t в ячейке с номером i содержится символ a_j внешнего алфавита машины Тьюринга T* .

Полагаем

$$\mathcal{B}_{i,t} \equiv U(C\langle i, 0, t \rangle, C\langle i, 1, t \rangle, \dots, C\langle i, m, t \rangle).$$

Содержательно формула $\mathcal{B}_{i,t}$ утверждает, что

" i -я клетка ленты в момент времени t содержит в точности один символ внешнего алфавита машины T ".

Напомним, что

$$A_T = \{ a_0, a_1, \dots, a_m \},$$

где A_T – внешний алфавит машины T .

Ясно, что длина формулы $\mathcal{B}_{i,t}$ есть $O(1)$.

Полагаем

$$\mathcal{B} \equiv \bigwedge_{\substack{0 \leq i \leq p(n) \\ 0 \leq t \leq p(n)}} \mathcal{B}_{i,t}.$$

Напомним, что

$$Q_T = \{ q_0, q_1, \dots, q_s \},$$

где Q_T – внутренний алфавит машины T .

Если мы считаем, что каждая пропозициональная переменная имеет длину 1, то длина формулы \mathcal{B} есть $O(p^2(n))$. При других предположениях длина формулы \mathcal{B} есть $O(p^3(n))$.

3) Построим формулу \mathcal{C} , содержательно утверждающую, что *в каждый момент времени $0 \leq t \leq p(n)$ машина Тьюринга T находится в точности в одном внутреннем состоянии*. Для построения формулы \mathcal{C} рассмотрим счетное множество пропозициональных переменных

$$\{ S\langle k, t \rangle \mid k, t = 0, 1, 2, \dots \}.$$

Содержательно истинность переменной $S\langle k, t \rangle$ означает, что *в момент времени t машина Тьюринга T находится в состоянии q_k* .

Полагаем

$$\mathcal{C}_t \equiv U(S\langle 0, t \rangle, S\langle 1, t \rangle, \dots, S\langle s, t \rangle), \quad \mathcal{C} \equiv \mathcal{C}_0 \& \mathcal{C}_1 \& \dots \& \mathcal{C}_{p(n)}.$$

Если мы считаем, что каждая пропозициональная переменная имеет длину 1, то длина формулы \mathcal{C} есть $O(p(n))$. При других предположениях длина формулы \mathcal{C} есть $O(p^2(n))$.

4) Построим формулу \mathcal{D} , содержательно утверждающую, что в каждый момент времени $0 \leq t \leq p(n)$ может измениться содержимое не более чем одной клетки.

Полагаем

$$\mathcal{D} \Rightarrow \bigwedge_{\substack{0 \leq i, t \leq p(n) \\ 0 \leq j \leq m}} \mathcal{D}_{i,j,t},$$

где

$$\mathcal{D}_{i,j,t} \Rightarrow (H\langle i, t \rangle \vee (C\langle i, j, t+1 \rangle \equiv C\langle i, j, t \rangle)),$$

где $X \equiv Y$ – это $(\overline{X} \vee Y) \& (X \vee \overline{Y})$.

Формула $\mathcal{D}_{i,j,t}$ содержательно утверждает, что

либо

а) в момент времени t головка обозревает ячейку с номером i ,

либо

б) в ячейке с номером i в момент времени $t+1$ записан j -й символ тогда и только тогда, когда в ячейке с номером i в момент времени t записан j -й символ.

Так как формула \mathcal{A} утверждает, что в каждый момент времени головка обозревает в точности одну ячейку, а формула \mathcal{B} утверждает, что в каждый момент времени каждая ячейка содержит единственный символ, то формула \mathcal{D} утверждает, что в каждый момент времени может измениться содержимое не более чем одной ячейки (обозреваемой).

Если мы считаем, что каждая пропозициональная переменная имеет длину 1, то длина формулы \mathcal{D} есть $O(p^2(n))$. При других предположениях длина формулы \mathcal{D} есть $O(p^3(n))$.

5) Построим формулу \mathcal{E} , содержательно утверждающую, что в каждый момент времени t следующая конфигурация K машины Тьюринга T получается из текущей переходом, разрешенным функцией переходов δ машины Тьюринга T .

Обозначим через E_{ijkt} дизъюнкцию следующих утверждений:

а) в момент времени t головка не обозревает клетку с номером i ,

б) в момент времени t клетка с номером i не содержит символ с номером j ,

в) в момент времени t машина Тьюринга T не находится в состоянии с номером k ,

г) следующая конфигурация машины Тьюринга T получается из текущей переходом, разрешенным функцией переходов δ машины Тьюринга T .

Тогда

$$\mathcal{E} = \bigwedge_{\substack{1 \leq k \leq s \\ 0 \leq j \leq m \\ 1 \leq i, t \leq p(n)}} E_{ijkt},$$

где

$$E_{ijkt} = \neg H\langle i, t \rangle \vee \neg C\langle i, j, t \rangle \vee \neg S\langle k, t \rangle \vee \bigvee_{\langle q_\alpha, a_\beta, D_\gamma \rangle \in \delta(q_k, a_j)} (C\langle i, \beta, t+1 \rangle \& S\langle \alpha, t+1 \rangle \& H\langle i+\gamma, t+1 \rangle),$$

где

$$D_{-1} = L, \quad D_0 = S, \quad D_{+1} = R.$$

Так как T – недетерминированная машина Тьюринга, то множество $\delta(q_k, a_j)$ может содержать несколько наборов $\langle q_\alpha, a_\beta, D_\gamma \rangle$, однако их число ограничено зависящей лишь от машины T константой. Поэтому нетрудно проверить, что если мы считаем, что каждая пропозициональная переменная имеет длину 1, то длина формулы \mathcal{E} есть $O(p^2(n))$. При других предположениях длина формулы \mathcal{E} есть $O(p^3(n))$.

6) По произвольному слову $W = a_{j_1} \dots a_{j_n}$ во внешнем алфавите машины T построим формулу \mathcal{F} , содержательно утверждающую, что *в начальный момент времени 0 машина находится в начальной конфигурации – ”в первых n ячейках ленты располагается слово W , а остальные $p(n) - n$ ячеек пусты (в них располагается пустой символ a_0), головка находится в начальном состоянии q_1 и обозревает самую первую ячейку”*

$$F = \bigwedge_{1 \leq i \leq n} C\langle i, j_i, 0 \rangle \& \bigwedge_{n < i \leq p(n)} C\langle i, 0, 0 \rangle \& S\langle 1, 0 \rangle \& H\langle 1, 0 \rangle.$$

И вновь нетрудно проверить, что если мы считаем, что каждая пропозициональная переменная имеет длину 1, то длина формулы \mathcal{E} есть $O(p(n))$. При других предположениях длина формулы \mathcal{E} есть $O(p^2(n))$.

7) Построим формулу \mathcal{G} , содержательно утверждающую, что *в некоторый момент времени $t \leq p(n)$ машина Тьюринга T пришла в заключительную конфигурацию.*

$$\mathcal{G} = S\langle 0, p(n) \rangle.$$

Обозначим через $\Psi(W)$ конъюнкцию построенных формул

$$\mathcal{A} \& \mathcal{B} \& \mathcal{C} \& \mathcal{D} \& \mathcal{E} \& \mathcal{F} \& \mathcal{G}.$$

Длина формулы $\Psi(W)$ есть $O(p^4(n))$. По исходному слову W формула $\Psi(W)$ строится за полиномиальное время.

Сделанные в ходе построения формулы $\Psi(W)$ пояснения показывают, что справедлива эквивалентность

слово W принадлежит языку $L(T)$, принимаемому недетерминированной машиной Тьюринга T тогда и только тогда, когда формула $\Psi(W)$ выполнима.

Таким образом, *проблема вхождения в произвольный язык из класса \mathcal{NP} полиномиально сводима к проблеме выполнимости для формул Логике высказываний (для булевых функций).* \square

Напомним, что формула F находится в *конъюнктивной нормальной форме* (КНФ), если она имеет вид

$$(\mathcal{A}_1 \& \dots \& \mathcal{A}_m),$$

где каждая формула \mathcal{A}_i является элементарной дизъюнкцией, т.е. имеет вид

$$(A_1^{\alpha_1} \vee \dots \vee A_n^{\alpha_n}),$$

причем $\alpha_j \in \{-1, 0, 1\}$ ($j = 1, \dots, n$) и A_j^{-1} – это отрицание $\overline{A_j}$ переменной A_j , A_j^1 – это просто сама переменная A_j , а A_j^0 – это пустое слово, т.е. в этом случае переменная A_j просто отсутствует.

Каждая переменная A_j и ее отрицание $\overline{A_j}$ называются *литералом*.

Формула вида

$$(A_1^{\alpha_1} \vee \dots \vee A_n^{\alpha_n}),$$

т.е. дизъюнкция литералов, называется *элементарной дизъюнкцией*, а формула вида

$$(A_1^{\alpha_1} \& \dots \& A_n^{\alpha_n}),$$

т.е. конъюнкция литералов, называется *элементарной конъюнкцией*.

Таким образом, конъюнктивная нормальная форма – конъюнкция элементарных дизъюнкций.

Двойственным понятием является понятие *дизъюнктивной нормальной формы* (ДНФ) – дизъюнкции элементарных конъюнкций.

Если при этом каждая элементарная дизъюнкция

$$(A_1^{\alpha_1} \vee \dots \vee A_n^{\alpha_n})$$

содержит не более k литералов, то формула F находится в *k -конъюнктивной нормальной форме* (k -КНФ).

Проблема k -выполнимости. По произвольной формуле, находящейся в k -КНФ, требуется определить, выполнима ли она.

Теорема 1.2. Проблема 3-выполнимости является \mathcal{NP} -полной.

До к а з а т е л ь с т в о. Прежде всего покажем, что каждая из построенных в ходе доказательства предыдущей теоремы формул \mathcal{A} , \mathcal{B} , \mathcal{C} , \mathcal{D} , \mathcal{E} , \mathcal{F} и \mathcal{G} либо уже находится в КНФ, либо может быть преобразована в КНФ, причем ее длина увеличится не более чем в C раз, где C – некоторая константа.

Формула \mathcal{U} находится в КНФ, поэтому в КНФ находятся и формулы \mathcal{A} , \mathcal{B} и \mathcal{C} .

Формулы \mathcal{F} и \mathcal{G} находятся в КНФ.

Для преобразования формулы \mathcal{D} в КНФ воспользуемся эквивалентностью

$$(Z \vee (X \equiv Y)) \sim ((Z \vee \overline{X} \vee Y) \& (Z \vee X \vee \overline{Y})).$$

Заметим, что

$$|((Z \vee \bar{X} \vee Y) \& (Z \vee X \vee \bar{Y}))| \leq 2 \cdot |(Z \vee (X \equiv Y))|.$$

Поэтому после преобразования формулы \mathcal{D} в КНФ ее длина увеличится не более чем в два раза.

Длина каждой формулы E_{ijkt} зависит лишь от рассматриваемой машины Тьюринга, точнее от ее функции переходов. Значит, можно преобразовать формулу \mathcal{E} в КНФ так, что ее длина увеличится в постоянное, не зависящее от $n = |W|$ число раз.

Выполнив указанные преобразования, мы преобразуем формулу $\Psi(W)$ в КНФ $\Psi'(W)$ и при этом ее длина увеличится в постоянное, не зависящее от n число раз.

Покажем, что проблема выполнимости формул в КНФ полиномиально сводима к проблеме **3-выполнимость**.

Покажем, что при $p \geq 4$ *элементарная дизъюнкция*

$$(X_1 \vee X_2 \vee \dots \vee X_p),$$

где X_1, \dots, X_p – литералы, выполнима тогда и только тогда, когда выполнима конъюнкция

$$(X_1 \vee X_2 \vee Y_1) \& (X_3 \vee \bar{Y}_1 \vee Y_2) \& (X_4 \vee \bar{Y}_2 \vee Y_3) \& \dots \\ \& (X_{p-2} \vee \bar{Y}_{p-4} \vee Y_{p-3}) \& (X_{p-1} \vee A_p \vee \bar{Y}_{p-3}),$$

где Y_1, \dots, Y_{p-3} – новые переменные.

Докажем, что набор $\langle \varepsilon_1, \dots, \varepsilon_p \rangle$ значений литералов A_1, \dots, A_p выполняет элементарную дизъюнкцию

$$(A_1 \vee A_2 \vee \dots \vee A_p)$$

тогда и только тогда, когда найдутся такие значения $\alpha_1, \dots, \alpha_{p-3}$ новых переменных Y_1, \dots, Y_{p-3} , что набор

$$\langle \varepsilon_1, \dots, \varepsilon_p, \alpha_1, \dots, \alpha_{p-3} \rangle$$

выполняет формулу

$$(X_1 \vee X_2 \vee Y_1) \& (X_3 \vee \bar{Y}_1 \vee Y_2) \& (X_4 \vee \bar{Y}_2 \vee Y_3) \& \dots \\ \& (X_{p-2} \vee \bar{Y}_{p-4} \vee Y_{p-3}) \& (X_{p-1} \vee X_p \vee \bar{Y}_{p-3}).$$

Если набор $\langle \varepsilon_1, \dots, \varepsilon_p \rangle$ значений литералов X_1, \dots, X_p выполняет элементарную дизъюнкцию

$$(X_1 \vee X_2 \vee \dots \vee X_p),$$

то найдется такое i , что $X_i = 1$. Полагаем $Y_j = 1$ при $j \leq i - 2$ и $Y_j = 0$ $j > i - 2$. Нетрудно понять, что построенный набор значений переменных выполняет формулу

$$(X_1 \vee X_2 \vee Y_1) \& (X_3 \vee \overline{Y_1} \vee Y_2) \& (X_4 \vee \overline{Y_2} \vee Y_3) \& \dots \\ \& (X_{p-2} \vee \overline{Y_{p-4}} \vee Y_{p-3}) \& (X_{p-1} \vee X_p \vee \overline{Y_{p-3}})$$

(истинность всех элементарных дизъюнкций кроме $(X_i \vee \overline{Y_{i-2}} \vee Y_{i-1})$ обеспечивается выбором значений переменных Y_1, \dots, Y_{i-3} , а истинность этой дизъюнкции следует из равенства $X_i = 1$).

Для доказательства обратного утверждения предположим, что набор

$$\langle \varepsilon_1, \dots, \varepsilon_p, \alpha_1, \dots, \alpha_{p-3} \rangle$$

выполняет формулу

$$(X_1 \vee X_2 \vee Y_1) \& (X_3 \vee \overline{Y_1} \vee Y_2) \& (X_4 \vee \overline{Y_2} \vee Y_3) \& \dots \\ \& (X_{p-2} \vee \overline{Y_{p-4}} \vee Y_{p-3}) \& (X_{p-1} \vee X_p \vee \overline{Y_{p-3}}).$$

Если $\alpha_1 = 0$, то $\varepsilon_1 = 1$ либо $\varepsilon_2 = 1$.

Если $\alpha_{p-3} = 1$, то $\varepsilon_{p-1} = 1$ либо $\varepsilon_p = 1$.

Если $\alpha_1 = 1$ и $\alpha_{p-3} = 0$, то найдется такое i ($1 \leq i \leq p - 4$), что $\alpha_i = 1$ и $\alpha_{i+1} = 0$.

Из равенства

$$(\varepsilon_{i+2} \vee \overline{\alpha_i} \vee \alpha_{i+1}) = 1$$

следует, что $\varepsilon_{i+2} = 1$. Поэтому набор $\langle \varepsilon_1, \dots, \varepsilon_p \rangle$ значений литералов X_1, \dots, X_p выполняет элементарную дизъюнкцию

$$(X_1 \vee \dots \vee X_p).$$

Выполнив указанные преобразования, мы преобразуем формулу $\Psi'(W)$, а значит и исходную формулу $\Psi(W)$, в 3-КНФ $\Phi_3(W)$, и при этом ее длина увеличится в постоянное, не зависящее от $n = |W|$ число раз.

Сделанные в ходе построения формул пояснения показывают, что справедлива эквивалентность

слово W принадлежит языку $L(T)$, принимаемому недетерминированной машиной Тьюринга T , тогда и только тогда, когда формула $\Psi_3(W)$ выполняется.

Таким образом, *проблема вхождения в произвольный язык из класса \mathcal{NP} полиномиально сводима к проблеме 3-КНФ выполнимости для формул Логике высказываний.* \square

Важным дополнением к доказанной теореме служит следующая теорема.

Теорема 1.3. *Проблема 2-выполнимости полиномиально разрешима.*

Построение полиномиального алгоритма может быть основано на следующих двух фактах:

1) из переменных x_1, \dots, x_n можно построить лишь $4n^2$ элементарных дизъюнкций вида $(x_i^\alpha \vee x_j^\beta)$,

2) если ни x_n , ни $\overline{x_n}$ не входят в формулы \mathcal{A} и \mathcal{B} , то формула

$$(x_n \vee \mathcal{A}) \& (\overline{x_n} \vee \mathcal{B})$$

выполнима тогда и только тогда, когда выполнима формула

$$(\mathcal{A} \vee \mathcal{B}).$$

Это позволяет вопрос о выполнимости набора элементарных дизъюнкций вида $(x_i^\alpha \vee x_j^\beta)$ от переменных x_1, \dots, x_n полиномиально свести к вопросу о выполнимости набора элементарных дизъюнкций такого же вида, но от переменных x_1, \dots, x_{n-1} . Продолжая аналогичным образом, мы дойдем до вопроса о выполнимости набора элементарных дизъюнкций вида $(x_i^\alpha \vee x_j^\beta)$ от переменных x_1 и x_2 , и все это можно выполнить за полиномиальное время.

§2. Сложность решения систем линейных уравнений

В этом параграфе будет установлена \mathcal{NP} -полнота некоторых проблем алгебры. Начнем мы с достаточно простых фактов – установим \mathcal{NP} -полноту проблем разрешимости систем линейных уравнений в числах 0, 1 (НОЛС-проблемы) и в целых неотрицательных числах (НЦЛС-проблемы). Принадлежность первой задачи классу \mathcal{NP} очевидна, а принадлежность второй будет установлена позже.

Покажем, что к каждой из этих проблем полиномиально сводима проблема 3-ВЫПОЛНИМОСТЬ.

Пропозициональные переменные будем обозначать через X_1, \dots, X_n и т.д.

Рассмотрим произвольную КНФ вида

$$\big\&_{i=1}^m \varphi_i, \quad \varphi_i = \bigvee_{j \in A_i} X_j^{\varepsilon_{i,j}}$$

где $\varepsilon_{i,j} \in \{-1, 1\}$, каждое множество A_i состоит из трех элементов, X_j^1 – это X_j , а X_j^{-1} – это $\neg X_j$.

Литералу X_j сопоставим переменную x_j , а литералу $\neg X_j$ – переменную x_j^{-1} .

Нетрудно понять, что формула

$$\big\&_{i=1}^m \varphi_i, \quad \varphi_i = \bigvee_{j \in A_i} X_j^{\varepsilon_{i,j}}$$

выполнима тогда и только тогда, когда имеет решение в неотрицательных целых числах следующая система линейных уравнений

$$\big\&_{i=1}^m \sum_{j \in A_i} x_j^{\varepsilon_{i,j}} - y_i = 1 \& \big\&_{j=1}^n x_j + x_j^{-1} = 1.$$

Так как при любом i выполняется неравенство

$$\sum_{j \in A_i} x_j^{\varepsilon_{i,j}} \leq 3,$$

то можно считать, что выполняются неравенства $0 \leq y_i \leq 2$. Поэтому формула

$$\bigwedge_{i=1}^m \varphi_i, \quad \varphi_i = \bigvee_{j \in A_i} X_j^{\varepsilon_{i,j}}$$

выполнима тогда и только тогда, когда имеет решение в числах 0 и 1 следующая система линейных уравнений

$$\sum_{i=1}^m \sum_{j \in A_i} x_j^{\varepsilon_{i,j}} - (y_{0,i} + y_{1,i}) = 1 \quad \& \quad \bigwedge_{j=1}^n x_j + x_j^{-1} = 1.$$

Достаточно неожиданным является следующий факт: рассмотренные проблемы остаются \mathcal{NP} -полными, даже если вместо системы линейных уравнений рассматривать одно линейное уравнение. За этими проблемами закрепились названия "Целочисленный рюкзак" и "0-1-рюкзак".

Целочисленный рюкзак. По произвольному уравнению вида

$$a_1 x_1 + \dots + a_n x_n = b,$$

где a_1, \dots, a_n и b – целые числа, определить, имеет ли оно неотрицательное целочисленное решение.

0-1-рюкзак. По произвольному уравнению вида

$$a_1 x_1 + \dots + a_n x_n = b,$$

где a_1, \dots, a_n и b – целые числа, определить, имеет ли оно решение в числах 0, 1.

Отметим, что вопрос о разрешимости в целых числах произвольного уравнения вида

$$a_1 x_1 + \dots + a_n x_n = b,$$

где a_1, \dots, a_n и b – целые числа, как хорошо известно из курса "Алгебра", решается за полиномиальное время.

Теорема 2.1. Проблема 0-1-рюкзак является \mathcal{NP} -полной.

Доказательство. Покажем, что вопрос о разрешимости в числах 0, 1 произвольной системы $Ax = b$ линейных уравнений полиномиально сводим к вопросу о разрешимости в числах 0, 1 одного линейного уравнения вида

$$a_1 x_1 + \dots + a_n x_n = c,$$

где a_1, \dots, a_n и c – целые числа.

Рассмотрим систему $Ax = b$ линейных уравнений с матрицей размера $m \times n$. Обозначим через A_1, \dots, A_m строки матрицы A , а через b_1, \dots, b_m — компоненты вектора b .

Пусть E — это наибольшее абсолютное значение элементов матрицы A и вектора b .

Для вектора x с компонентами 0 и 1 справедлива эквивалентность

$$\begin{aligned} A_1 \cdot x = b_1 \ \& \ \dots \ \& \ A_m \cdot x = b_m \iff \\ (A_1 + (2nE)A_2 + (2nE)^2A_3 + \dots + (2nE)^{m-1}A_m) \cdot x = \\ b_1 + (2nE)b_2 + (2nE)^2b_3 + \dots + (2nE)^{m-1}b_m. \end{aligned}$$

Это завершает доказательство теоремы. \square

Рассматриваемая задача остается \mathcal{NP} -полной, если даже считать, что все коэффициенты уравнения

$$a_1 x_1 + \dots + a_n x_n = b$$

положительны.

Рассмотрим уравнение с целыми коэффициентами вида

$$a_1 x_1 + \dots + a_n x_n = b,$$

где $b \geq 0$. Без ограничения общности можно считать, что $a_1 > 0, \dots, a_p > 0, a_{p+1} < 0, \dots, a_n < 0$. Рассматриваемое уравнение равносильно уравнению

$$\begin{aligned} a_1 x_1 + \dots + a_p x_p + (-a_{p+1})(-x_{p+1} + 1) + \dots + (-a_n)(-x_n + 1) = \\ b + (-a_{p+1}) + \dots + (-a_n). \end{aligned}$$

Последнее уравнение имеет решение в числах 0, 1 тогда и только тогда, когда имеет решение в числах 0, 1 следующее уравнение с положительными коэффициентами

$$a_1 y_1 + \dots + a_p y_p + (-a_{p+1})y_{p+1} + \dots + (-a_n)y_n = b + (-a_{p+1}) + \dots + (-a_n).$$

Проблема разрешимости в числах 0, 1 произвольного линейного уравнения вида

$$a_1 x_1 + \dots + a_n x_n = b,$$

где a_1, \dots, a_n и b — натуральные числа, носит название **задача об укладке ранца** или **задача об укладке рюкзака**. \mathcal{NP} -трудность этой задачи была использована Р. Мерклом и М. Хеллманом при разработке одного из первых алгоритмов шифрования с открытым ключом.

Прежде всего отметим, что в одном частном случае вопрос о разрешимости в числах 0, 1 рассматриваемого линейного уравнения тривиально разрешим. Речь идет о так называемом **сверхвозрастающем рюкзаке**. Такое название

закрепилось за задачей о разрешимости в числах 0, 1 произвольного линейного уравнения вида

$$a_1 x_1 + \dots + a_n x_n = b,$$

где a_1, \dots, a_n и b – натуральные числа, удовлетворяющие условию

$$\sum_{j=1}^i a_j < a_{i+1}, \quad i = 1, \dots, n-1.$$

Легко понять, что при $n > 1$ выбор значения x_n определяется условиями: если $a_n \leq b$, то $x_n = 1$, а если $a_n > b$, то $x_n = 0$. А при $n = 1$ вопрос решается тривиально.

В алгоритме Меркла – Хеллмана **закрытый ключ** включает в свой состав: сверхвозрастающую последовательность a_1, \dots, a_n натуральных чисел, натуральные числа M , t и t^{-1} такие, что

$$\sum_{j=1}^n a_j < M, \quad t \cdot t^{-1} \equiv 1 \pmod{M}.$$

Открытый ключ – это последовательность r_1, \dots, r_n остатков от деления чисел $t \cdot a_1, \dots, t \cdot a_n$ на число M .

Шифром слова $\varepsilon_1 \dots \varepsilon_n$, где каждое ε_i – это 0 или 1, служит число r

$$r = r_1 \varepsilon_1 + \dots + r_n \varepsilon_n.$$

Для расшифрования обладателю закрытого ключа достаточно решить следующую задачу о сверхвозрастающем рюкзаке

$$a_1 x_1 + \dots + a_n x_n = b,$$

где b – это остаток от деления числа $r \cdot t^{-1}$ на число M .

Алгоритм шифрования Меркла – Хеллмана нашел практические применения, был запатентован в ряде стран, однако "продержался" менее десяти лет – в конце 1980-х годов он был взломан. Та же участь постигла и ряд других алгоритмов шифрования, основанных на задаче об укладке ранца. Не вдаваясь в подробности, поясним, что хотя задача об укладке произвольного рюкзака и является \mathcal{NP} -трудной, таковой может не быть подзадача, используемая в алгоритме шифрования Меркла – Хеллмана, так как в ней рассматривается более узкий класс уравнений – линейные уравнения вида

$$r_1 x_1 + \dots + r_n x_n = r,$$

где последовательность r_1, \dots, r_n коэффициентов – это последовательность остатков от деления чисел $t \cdot a_1, \dots, t \cdot a_n$ на число M , причем последовательность a_1, \dots, a_n является сверхвозрастающей. Можно это выразить и по-другому, *последовательность a_1, \dots, a_n остатков от деления чисел $t^{-1} \cdot r_1, \dots, t^{-1} \cdot r_n$*

на число M является сверхвозрастающей. При такой формулировке уже трудно надеяться, что используемая ограниченная подзадача является достаточно трудной.

Установим \mathcal{NP} -полноту проблемы **Целочисленный рюкзак**.

Теорема 2.2. \mathcal{NP} -полной является задача определения по произвольному уравнению вида

$$a_1 x_1 + \dots + a_n x_n = b,$$

где a_1, \dots, a_n и b — целые числа, имеет ли оно неотрицательное целочисленное решение.

Доказательство. То, что рассматриваемая задача находится в классе \mathcal{NP} , будет установлено позже.

Покажем, что к ней полиномиально сводима проблема разрешимости в числах 0, 1 произвольного уравнения с натуральными коэффициентами.

Рассмотрим уравнение

$$a_1 x_1 + \dots + a_n x_n = b$$

с натуральными коэффициентами.

Если $b > a_1 + \dots + a_n$, то рассматриваемое уравнение не имеет решения в числах 0, 1.

При $b = a_1 + \dots + a_n$ такое решение есть.

Остается рассмотреть случай, когда $b < a_1 + \dots + a_n$.

Обозначим через s наименьшее натуральное число такое, что

$$a_1 + \dots + a_n < 2^s,$$

т.е. $s = \lceil \log_2(a_1 + \dots + a_n) \rceil + 1$.

Рассмотрим уравнение

$$\begin{aligned} & (2^0 + 2^{3ns} a_1) \cdot u_1 + (2^{3s} + 2^{3ns} a_2) \cdot u_2 + \dots \\ & \dots + (2^{3(i-1)s} + 2^{3is} a_i) \cdot u_i + \dots \\ & (2^{3(n-1)s} + 2^{3ns} a_n) \cdot u_n + \\ & (2^0 + 2^{(3n+1)s} a_1) \cdot v_1 + (2^{3s} + 2^{3(n+1)s} a_2) \cdot v_2 + \dots \\ & \dots + (2^{3(i-1)s} + 2^{3(i+1)s} a_i) \cdot v_i + \dots \\ & (2^{3(n-1)s} + 2^{3(n+1)s} a_n) \cdot v_n = \\ & 2^0 + 2^{3s} + \dots + 2^{3(i-1)s} + \dots + 2^{3(n-1)s} + 2^{3ns} b + 2^{(3n+1)s} (A - b), \end{aligned}$$

где $A = a_1 + \dots + a_n$.

Покажем, что уравнение

$$a_1 x_1 + \dots + a_n x_n = b$$

имеет решение в числах 0, 1 тогда и только тогда, когда построенное уравнение имеет неотрицательное решение.

Если набор $(\alpha_1, \dots, \alpha_n)$ чисел 0, 1 является решением первого уравнения

$$a_1 x_1 + \dots + a_n x_n = b,$$

набор

$$(\alpha_1, \dots, \alpha_n, 1 - \alpha_1, \dots, 1 - \alpha_n)$$

является неотрицательным целочисленным решением второго уравнения (даже его 0, 1-решением).

Для доказательства обратной импликации предположим, что набор неотрицательных целых чисел

$$(\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n)$$

является решением второго уравнения, т.е. выполнено равенство

$$\begin{aligned} & (2^0 + 2^{3ns} a_1) \cdot \alpha_1 + (2^{3s} + 2^{3ns} a_2) \cdot \alpha_2 + \dots \\ & \dots + (2^{3(i-1)s} + 2^{3is} a_i) \cdot \alpha_i + \dots \\ & (2^{3(n-1)s} + 2^{3ns} a_n) \cdot \alpha_n + \\ & (2^0 + 2^{(3n+1)s} a_1) \cdot \beta_1 + (2^{3s} + 2^{(3n+1)s} a_2) \cdot \beta_2 + \dots \\ & \dots + (2^{3(i-1)s} + 2^{(3i+1)s} a_i) \cdot \beta_i + \dots \\ & (2^{3(n-1)s} + 2^{(3n+1)s} a_n) \cdot \beta_n = \\ & 2^0 + 2^{3s} + \dots + 2^{3(i-1)s} + \dots + 2^{3(n-1)s} + 2^{3ns} b + 2^{(3n+1)s} (A - b). \end{aligned}$$

Обозначим через L левую часть предыдущего равенства, а через R – его правую часть.

Индукцией по i докажем, что $\alpha_i + \beta_i = 1$.

Для доказательства равенства $\alpha_1 + \beta_1 = 1$, перейдя к сравнению по $\text{mod } 2^{3s}$, получим

$$\alpha_1 + \beta_1 \equiv 1 \pmod{2^{3s}}.$$

Если $\alpha_1 + \beta_1 \neq 1$, то $\alpha_1 + \beta_1 > 2^{3s}$, что дает неравенство

$$\begin{aligned} L & \geq (2^0 + 2^{3ns} a_1) \alpha_1 + (2^0 + 2^{(3n+1)s} a_1) \beta_1 > \\ & (2^0 + 2^{3ns} a_1) (\alpha_1 + \beta_1) > 2^{3s} + 2^{3ns} \cdot 2^{3s} > R, \end{aligned}$$

что противоречит равенству $L = R$.

Предположим, что уже доказаны равенства

$$\alpha_1 + \beta_1 = 1 \ \& \ \dots \ \& \ \alpha_{i-1} + \beta_{i-1} = 1.$$

Перейдя к сравнению по $\text{mod } 2^{3is}$, получим

$$2^{3(i-1)s} (\alpha_i + \beta_i) \equiv 2^{3(i-1)s} \pmod{2^{3is}}.$$

Значит,

$$\alpha_i + \beta_i \equiv 1 \pmod{2^{3s}}.$$

Если $\alpha_i + \beta_i \neq 1$, то $\alpha_i + \beta_i > 2^{3s}$, что по той же схеме дает противоречивое неравенство $L > R$.

Значит, выполнены равенства

$$\alpha_1 + \beta_1 = 1 \text{ \& } \dots \text{ \& } \alpha_n + \beta_n = 1,$$

поэтому набор неотрицательных целых чисел

$$(\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_n)$$

является 0,1-решением второго уравнения и выполнено равенство

$$2^{3ns}(a_1\alpha_1 + \dots + a_n\alpha_n) + 2^{(3n+1)s}(a_1\beta_1 + \dots + a_n\beta_n) = 2^{3ns}b + 2^{(3n+1)s}(A - b).$$

Заменив β_i на $1 - \alpha_i$, получим

$$2^{3ns}(1 - 2^s)(a_1\alpha_1 + \dots + a_n\alpha_n) = 2^{3ns}(1 - 2^s)b, \quad a_1\alpha_1 + \dots + a_n\alpha_n = b.$$

Таким образом набор $(\alpha_1, \dots, \alpha_n)$ чисел 0, 1 является решением первого уравнения. \square

Ближайшей нашей целью является доказательство принадлежности классу \mathcal{NP} рассмотренных выше проблем для целочисленных систем линейных уравнений.

Предварительно приведем некоторые необходимые для дальнейшего изложения факты, относящиеся к системам линейных неравенств. Изучение систем линейных неравенств одним из первых выполнил немецкий математик Герман Минковский (1864 – 1909) в конце XIX века. Им был получен ряд фундаментальных результатов в этой области.

Первое утверждение, которое нам потребуется – это *теорема о следствиях совместных систем линейных уравнений*.

Пусть $A = ||a_{ij}||$ – матрица размера $m \times n$, $b = [b_1, \dots, b_m]$ – вектор-столбец высоты m , $c = (c_1, \dots, c_n)$ – вектор-строка длины n с элементами из произвольного поля F , а d – элемент этого поля.

Теорема 2.3. *Если каждое решение совместной системы линейных уравнений $Ax = b$ в поле F является решением уравнения $sx = d$, то вектор (c, d) представим в виде линейной комбинации строк матрицы (A, b) .*

До к а з а т е л ь с т в о. Обозначим через r ранг матрицы A . Из равносильности систем

$$Ax = b \quad \text{и} \quad \begin{pmatrix} A \\ c \end{pmatrix} \cdot x = \begin{pmatrix} b \\ d \end{pmatrix}$$

следует совпадение их многообразий решений, а значит, и совпадение пространств V и U решений соответствующих однородных систем

$$Ax = 0 \quad \text{и} \quad \begin{pmatrix} A \\ c \end{pmatrix} \cdot x = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

Последнее влечет равенство рангов

$$r = \text{rank } A = \text{rank} \begin{pmatrix} A \\ c \end{pmatrix}.$$

Из совместности систем

$$Ax = b \quad \text{и} \quad \begin{pmatrix} A \\ c \end{pmatrix} \cdot x = \begin{pmatrix} b \\ d \end{pmatrix}$$

по теореме Кронекера – Капелли получаем равенства

$$\begin{aligned} r &= \text{rank } A = \text{rank } (A, b), \\ r &= \text{rank} \begin{pmatrix} A \\ c \end{pmatrix} = \text{rank} \begin{pmatrix} A & b \\ c & d \end{pmatrix}. \end{aligned}$$

Значит,

$$r = \text{rank } (A, b) = \text{rank} \begin{pmatrix} A & b \\ c & d \end{pmatrix}.$$

Поэтому строка (c, d) является линейной комбинацией строк матрицы (A, b) . \square

Обозначим через K поле рациональных Q или действительных чисел R .

На множестве K_n векторов-столбцов высоты n введем отношение частичного порядка \geq , полагая:

$$b = [b_1, \dots, b_n] \geq c = [c_1, \dots, c_n] \iff b_1 \geq c_1 \ \& \ \dots \ \& \ b_n \geq c_n.$$

Аналогичным образом вводится отношение частичного порядка \geq и на множестве K^n векторов-строк длины n .

Вектор d называется *неотрицательным*, если $d \geq 0$, где 0 – нулевой вектор. Другими словами, вектор d *неотрицателен* тогда и только тогда, когда *неотрицательны все его компоненты*.

Пусть $A = ||a_{ij}||$ – матрица размера $m \times n$, а $b = [b_1, \dots, b_m]$ – вектор-столбец высоты m с элементами из поля K .

Рассмотрим *вопрос о разрешимости системы уравнений* $Ax = b$ *в неотрицательных числах из поля* K , т.е. вопрос о разрешимости системы уравнений и неравенств

$$\begin{aligned} a_{11}x_1 + \dots + a_{1n}x_n &= b_1 \\ \dots &\dots \\ a_{m1}x_1 + \dots + a_{mn}x_n &= b_m \\ x_1 \geq 0 \ \& \ \dots \ \& \ x_n &\geq 0. \end{aligned}$$

Будем считать, что рассматриваемая система совместна, т.е. выполняется условие Кронекера – Капелли равенства рангов

$$\text{rank } A = \text{rank } (A|b).$$

Докажем вспомогательные леммы, которые позволят нам оценить величину наименьшего решения рассматриваемой системы. Можно считать, что *строки матрицы A линейно независимы*, поэтому $m \leq n$ и *ранг матрицы A равен m* . В случае, когда $m = n$, решение системы $Ax = b$ единственно и может быть найдено либо методом Гаусса, либо по формулам Крамера. Интерес представляет случай, когда $m < n$.

Для произвольной базы $\mathcal{B} = \{A^{(j_1)}, \dots, A^{(j_m)}\}$ системы столбцов матрицы A , т.е. для любых ее $m = \text{rank } A$ линейно независимых столбцов, определяется *базисное или опорное решение*, которое строится следующим образом:

система $Ax = b$ заменяется равносильной ей системой

$$A^{(j_1)}x_{j_1} + \dots + A^{(j_m)}x_{j_m} = b - \sum_{j \neq j_1, \dots, j_m} A^{(j)}x_j,$$

которую запишем в виде

$$B[x_{j_1}, \dots, x_{j_m}] = b - \sum_{j \neq j_1, \dots, j_m} A^{(j)}x_j.$$

Ее решение имеет вид

$$[x_{j_1}, \dots, x_{j_m}] = B^{-1}(b - \sum_{j \neq j_1, \dots, j_m} A^{(j)}x_j).$$

Поэтому

$$\begin{aligned} x_{j_1} &= c_1 + \sum_{j \neq j_1, \dots, j_m} c_{1j}x_j \\ &\dots \dots \dots \\ x_{j_m} &= c_m + \sum_{j \neq j_1, \dots, j_m} c_{mj}x_j. \end{aligned}$$

Переменные x_j при $j \neq j_1, \dots, j_m$ называются *свободными* переменными, а остальные – *базисными*. Полагая равными нулю свободные переменные и вычисляя с использованием предыдущих равенств значения базисных переменных, получим *базисное или опорное решение*.

Лемма 3. *Если система линейных уравнений*

$$\begin{aligned} a_{11}x_1 + \dots + a_{1n}x_n &= b_1 \\ \dots \dots \dots \\ a_{m1}x_1 + \dots + a_{mn}x_n &= b_m \end{aligned}$$

над полем K разрешима в неотрицательных числах, то у нее имеется неотрицательное базисное (опорное) решение.

Д о к а з а т е л ь с т в о. Обозначим через $(\alpha_1, \dots, \alpha_n)$ решение рассматриваемой системы в неотрицательных числах из поля K с наибольшим возможным числом нулевых компонент. Если все α_i ($i = 1, \dots, n$) равны нулю, то $b = 0$ и рассматриваемое нулевое решение однородной системы – опорное.

Чтобы несколько упростить обозначения без ограничения общности можно считать, что в решении отличны от нуля первые k компонент, а остальные равны нулю. Значит $\alpha_1 > 0, \dots, \alpha_k > 0$. Тогда выполняется равенство

$$\alpha_1 \cdot A^{(1)} + \dots + \alpha_k \cdot A^{(k)} = b$$

Покажем, что столбцы $A^{(1)}, \dots, A^{(k)}$ матрицы A линейно независимы. Предположим противное, пусть β_1, \dots, β_k – отличный от нулевого набор действительных чисел такой, что

$$\beta_1 \cdot A^{(1)} + \dots + \beta_k \cdot A^{(k)} = 0.$$

Без ограничения общности можно считать, что $\beta_1 \neq 0$ и даже $\beta_1 > 0$.

Без ограничения общности можно считать, что $\beta_k \cdot (\alpha_k)^{-1}$ – наибольший элемент среди чисел

$$\beta_1 \cdot (\alpha_1)^{-1}, \dots, \beta_k \cdot (\alpha_k)^{-1}.$$

Так как

$$0 < \beta_1 \cdot (\alpha_1)^{-1} \leq \beta_k \cdot (\alpha_k)^{-1},$$

то $\beta_k \cdot (\alpha_k)^{-1} > 0$. Но $\alpha_k > 0$, поэтому и $\beta_k > 0$.

Рассмотрим равенства

$$b = b - 0 =$$

$$\begin{aligned} \sum_{i=1}^k \alpha_i \cdot A^{(i)} - \left(\sum_{i=1}^k \beta_i \cdot A^{(i)} \right) \cdot (\beta_k)^{-1} \cdot \alpha_k = \\ \sum_{i=1}^k A^{(i)} \cdot (\alpha_i - \beta_i \cdot (\beta_k)^{-1} \cdot \alpha_k). \end{aligned}$$

Полагаем $\gamma_i = \alpha_i - \beta_i \cdot (\beta_k)^{-1} \cdot \alpha_k$.

Заметим, что $\gamma_k = \alpha_k - \beta_k \cdot (\beta_k)^{-1} \cdot \alpha_k = 0$, поэтому выполняется равенство

$$b = \sum_{i=1}^{k-1} A^{(i)} \cdot \gamma_i.$$

Кроме того

$$\gamma_i = \alpha_i - \beta_i \cdot (\beta_k)^{-1} \cdot \alpha_k = \alpha_i \cdot (1 - (\alpha_i)^{-1} \cdot \beta_i \cdot (\beta_k)^{-1} \cdot \alpha_k) \geq 0.$$

Тогда набор $(\gamma_1, \dots, \gamma_{k-1}, 0, 0, \dots, 0)$ – решение рассматриваемой системы в неотрицательных действительных числах с большим числом нулевых компонент, чем решение $(\alpha_1, \dots, \alpha_n)$. Что противоречит предположению $(\alpha_1, \dots, \alpha_n)$ – решение рассматриваемой системы в неотрицательных действительных числах с наибольшим возможным числом нулевых компонент. Поэтому столбцы $A^{(1)}, \dots, A^{(k)}$ матрицы A линейно независимы.

Дополним столбцы $A^{(1)}, \dots, A^{(k)}$ до базы системы столбцов. Чтобы не усложнять обозначения, можно считать, что полученную базу образуют первые m столбцов. Равенство

$$\alpha_1 \cdot A^{(1)} + \dots + \alpha_k \cdot A^{(k)} + 0 \cdot A^{(k+1)} + \dots + 0 \cdot A^{(m)} = b - 0 \cdot A^{(m+1)} - \dots - 0 \cdot A^{(n)}$$

показывает, что $(\alpha_1, \dots, \alpha_n)$ – базисное (опорное) решение рассматриваемой системы в неотрицательных числах из поля K . \square

Рассмотрим систему линейных неравенств

$$\begin{aligned} a_{11}x_1 + \dots + a_{1n}x_n &\leq b_1 \\ \dots &\dots \\ a_{m1}x_1 + \dots + a_{mn}x_n &\leq b_m \end{aligned} \tag{1}$$

над полем K , которую будем записывать в виде $Ax \leq b$.

Введем новые переменные y_1, \dots, y_m и рассмотрим систему линейных уравнений

$$\begin{aligned} a_{11}x_1 + \dots + a_{1n}x_n + y_1 &= b_1 \\ \dots &\dots \\ a_{m1}x_1 + \dots + a_{mn}x_n + y_m &= b_m \end{aligned} \tag{2}$$

над полем K , которую будем записывать в виде

$$(A, E_m) \cdot \begin{pmatrix} x \\ y \end{pmatrix} = b.$$

Легко понять, что вектор-столбец s высоты n является решением системы неравенств (1) тогда и только тогда, когда найдется неотрицательный вектор-столбец d высоты m такой, что вектор-столбец $\begin{pmatrix} s \\ d \end{pmatrix}$ является решением системы уравнений (2). Это позволяет свести вопрос о разрешимости данной системы неравенств к вопросу о существовании неотрицательного решения y некоторой системы линейных уравнений.

Если каждое решение над полем K системы линейных неравенств $Ax \leq b$ является решением неравенства $sx \leq d$, то неравенство $sx \leq d$ называется следствием системы линейных неравенств $Ax \leq b$.

Легко понять, что если вектор s является линейной комбинацией с неотрицательными коэффициентами системы строк матрицы A , то неравенство $sx \leq 0$ является следствием системы неравенств $Ax \leq 0$. Справедливо и обратное утверждение, которое называется теоремой Минковского.

Теорема 2.4 (Минковский). Если неравенство $sx \leq 0$ является следствием системы неравенств $Ax \leq 0$, то вектор s является линейной комбинацией с неотрицательными коэффициентами системы строк матрицы A .

Докажем вспомогательную лемму.

Лемма 4. Если неравенство $sx \leq 0$ является следствием системы неравенств

$$\begin{pmatrix} A_1 \\ \dots \\ A_{m-1} \\ A_m \end{pmatrix} \cdot x \leq 0$$

и

$$c = \lambda_1 A_1 + \dots + \lambda_m A_m,$$

причем $\lambda_1, \dots, \lambda_{m-1} \geq 0$ и $\lambda_m < 0$, то неравенство $sx \leq 0$ является следствием системы неравенств

$$\begin{pmatrix} A_1 \\ \dots \\ A_{m-1} \end{pmatrix} \cdot x \leq 0.$$

Доказательство. Пусть α – решение последней системы неравенств. Если $A_m \alpha \leq 0$, то α – решение исходной системы неравенств, поэтому $\alpha \leq 0$.

Если же $A_m \alpha > 0$, то вновь

$$s\alpha = \lambda_1 A_1 \alpha + \dots + \lambda_{m-1} A_{m-1} \alpha + \lambda_m A_m \alpha \leq 0.$$

□

Доказательство теоремы Минковского. Покажем, что уравнение $sx = 0$ является следствием системы уравнений $Ax = 0$.

Если $Ad = 0$, то и $A(-d) = 0$, поэтому $Ad \leq 0$ и $A(-d) \leq 0$, а значит, $cd \leq 0$ и $c(-d) \leq 0$, следовательно, $cd = 0$. Значит, вектор c является линейной комбинацией системы строк матрицы A

$$c = \lambda_1 A_1 + \dots + \lambda_m A_m.$$

Остается показать, что вектор c является линейной комбинацией с неотрицательными коэффициентами системы строк матрицы A .

Если A – нулевая матрица, то $c = 0$, и утверждение очевидно.

Пусть A – ненулевая матрица. Доказательство проведем индукцией по числу m неравенств в системе.

При $m = 1$ система $Ax \leq 0$ принимает вид

$$a_{11}x_1 + \dots + a_{1n}x_n \leq 0.$$

Без ограничения общности можно считать, что $a_{11} \neq 0$.

$$c = \lambda_1 A_1 = \lambda_1(a_{11}, \dots, a_{1n}).$$

Если $a_{11} > 0$, то вектор $(-1, 0, \dots, 0)$, являющийся решением системы неравенств $Ax \leq 0$, будет решением и ее следствия $sx \leq 0$, поэтому $\lambda_1 a_{11} \geq 0$, значит $\lambda_1 \geq 0$.

Если $a_{11} < 0$, то вектор $(1, 0, \dots, 0)$, являющийся решением системы неравенств $Ax \leq 0$, будет решением и ее следствия $cx \leq 0$, поэтому $\lambda_1 a_{11} \leq 0$, значит, $\lambda_1 \geq 0$.

Предположим, что утверждение доказано для произвольных однородных систем линейных неравенств с не более чем $m - 1$ неравенствами.

Рассмотрим систему линейных неравенств $Ax \leq 0$, где A – матрица размера $m \times n$.

Если неравенство $cx \leq 0$ является следствием этой системы, то

$$c = \lambda_1 A_1 + \dots + \lambda_m A_m = (\lambda_1, \dots, \lambda_m) A.$$

Среди всех равенств указанного вида, рассмотрим равенство с наибольшим числом t неотрицательных коэффициентов λ_i . Без ограничения общности можно считать, что неотрицательны первые t коэффициентов λ_i , т.е.

$$\lambda_1 \geq 0 \& \dots \& \lambda_t \geq 0.$$

Если $t = m$, то утверждение доказано.

Предположим, что $t < m$. Полагаем

$$g = \lambda_1 A_1 + \dots + \lambda_t A_t + \lambda_m A_m.$$

Покажем, что неравенство $gx \leq 0$ является следствием системы неравенств $Ax \leq 0$.

Если $A\gamma \leq 0$, то $c\gamma \leq 0$. Кроме того

$$(c - g) \cdot \gamma = \sum_{t < i < m} \lambda_i (A_i \gamma) \geq 0,$$

поэтому $g \cdot \gamma = c \cdot \gamma + (g - c) \cdot \gamma \leq 0$.

Но

$$g = \lambda_1 A_1 + \dots + \lambda_t A_t + 0 A_{t+1} + \dots + 0 A_{m-1} + \lambda_m A_m,$$

значит, по предыдущей лемме, неравенство $gx \leq 0$ является следствием системы неравенств

$$\begin{pmatrix} A_1 \\ \dots \\ A_{m-1} \end{pmatrix} \cdot x \leq 0.$$

По индуктивному предположению для некоторых неотрицательных чисел $\delta_1, \dots, \delta_{m-1}$ выполняется равенство

$$g = \sum_{i=1}^{m-1} \delta_i A_i.$$

Получаем равенства

$$\begin{aligned} c &= (c - g) + g = \sum_{t < i < m} \lambda_i (A_i \gamma) + \sum_{i=1}^{m-1} \delta_i A_i = \\ &= \delta_1 A_1 + \dots + \delta_t A_t + \\ &+ (\lambda_{t+1} + \delta_{t+1}) A_{t+1} + \dots + (\lambda_{m-1} + \delta_{m-1}) A_{m-1} + 0 A_m, \end{aligned}$$

которые дают представление вектора c в виде линейной комбинации строк матрицы A с более чем t неотрицательными коэффициентами. Что противоречит выбору числа t . Значит $t = m$. \square

Следующая теорема дает критерий несовместности системы линейных неравенств.

Теорема 2.5 (Критерий несовместности систем линейных неравенств).
Система линейных неравенств $Ax \leq b$ несовместна тогда и только тогда, когда следующая система линейных уравнений

$$\begin{pmatrix} A^t \\ b^t \end{pmatrix} \cdot y = \begin{pmatrix} 0 \\ \dots \\ 0 \\ -1 \end{pmatrix}$$

имеет неотрицательное решение.

Доказательство. Предположим, что λ – неотрицательное решение последней системы. Тогда $A^t \cdot \lambda = 0$, $b^t \cdot \lambda = -1$. Если c – решение системы линейных неравенств $Ax \leq b$, то $A \cdot c \leq b$, $c^t A^t \leq b^t$, $c^t A^t \lambda \leq b^t \lambda$, что дает противоречие $0 \leq -1$.

Предположим, что система линейных неравенств $Ax \leq b$ несовместна.

Введем в рассмотрение вспомогательную однородную систему неравенств

$$B \cdot y = (A, -b) \cdot \begin{pmatrix} x \\ x_{n+1} \end{pmatrix} \leq 0.$$

Пусть $c = [c_1, \dots, c_n, c_{n+1}]$ – произвольное решение этой системы неравенств. Если бы выполнялось неравенство $c_{n+1} > 0$, то вектор

$$[c_1/c_{n+1}, \dots, c_n/c_{n+1}]$$

был бы решением несовместной системы линейных неравенств $Ax \leq b$. Поэтому $c_{n+1} \leq 0$, т.е. неравенство

$$0x_1 + \dots + 0x_n + x_{n+1} \leq 0$$

является следствием системы неравенств

$$(A, -b) \cdot \begin{pmatrix} x \\ x_{n+1} \end{pmatrix} \leq 0.$$

По теореме Минковского найдутся такие неотрицательные числа $\lambda_1, \dots, \lambda_m$, для которых выполняется равенство

$$(0, \dots, 0, 1) = \sum_{i=1}^m D_i \lambda_i,$$

где D_1, \dots, D_m – это строки матрицы $D = (A, -b)$, т.е. равенство

$$\begin{pmatrix} A^t \\ -b^t \end{pmatrix} \cdot \lambda = \begin{pmatrix} 0 \\ \dots \\ 0 \\ 1 \end{pmatrix},$$

а значит, и равенство

$$\begin{pmatrix} A^t \\ b^t \end{pmatrix} \cdot \lambda = \begin{pmatrix} 0 \\ \dots \\ 0 \\ -1 \end{pmatrix}.$$

Следовательно, $\lambda_1, \dots, \lambda_m$ – неотрицательное решение системы линейных уравнений

$$\begin{pmatrix} A^t \\ b^t \end{pmatrix} \cdot y = \begin{pmatrix} 0 \\ -1 \end{pmatrix}.$$

□

Пусть $A = ||a_{ij}||$ – целочисленная матрица размера $m \times n$, а $b = [b_1, \dots, b_m]$ – целочисленный вектор-столбец высоты m . Введем обозначения

$$C = \max_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} |a_{ij}|, \quad D = \max_{1 \leq i \leq m} |b_i|, \quad C_1 = (mC)^{m+1}.$$

Лемма 5. Если $x = (x_1, \dots, x_n)$ – базисное решение системы линейных уравнений $Ax = b$, то при любом $j = 1, \dots, x_n$ выполняется неравенство

$$|x_j| \leq m! C^{m-1} D.$$

Доказательство. Свободные переменные в базисном решении равны нулю, поэтому для них неравенство очевидно.

Базисные переменные определяются равенством

$$[x_{j_1}, \dots, x_{j_m}] = B^{-1}b.$$

Рассмотрим матрицу $B = ||b_{ts}||$. Напомним, что $B^{-1} = 1/\det(B) \cdot ||B_{st}||$. Так как B_{st} – это минор порядка $m-1$ матрицы B , то

$$|B_{st}| \leq (m-1)! \cdot C^{m-1}.$$

Но $\det(B) \geq 1$, поэтому из равенства

$$x_{j_t} = 1/\det(B) \cdot \sum_{s=1}^m B_{st} \cdot b_s$$

получаем

$$|x_{j_t}| \leq 1/\det(B) \cdot \sum_{s=1}^m |B_{st}| \cdot |b_s| \leq$$

$$m \cdot (m-1)! \cdot C^{m-1} \cdot D = m! \cdot C^{m-1} \cdot D.$$

□

Лемма 6. Для векторов-столбцов u_1, \dots, u_p высоты t с целочисленными компонентами, не превосходящими по модулю числа E , следующие условия эквивалентны:

1) существуют неотрицательные действительные числа $\alpha_1, \dots, \alpha_p$, не все равные нулю, для которых выполнено равенство

$$\alpha_1 u_1 + \dots + \alpha_p u_p = 0;$$

2) существуют неотрицательные целые числа $\alpha_1, \dots, \alpha_p$, не превосходящие числа $(tE)^{m+1}$ и не все равные нулю, для которых выполнено равенство

$$\alpha_1 u_1 + \dots + \alpha_p u_p = 0;$$

3) не существует вектора-столбца h высоты t с действительными компонентами такого, что при любом j ($j = 1, \dots, p$) $p_j = h^t \cdot u_j > 0$,

4) не существует вектора-столбца h высоты t с целыми компонентами, не превосходящими по модулю числа $(tE)^{m+1}$, такого, что при любом j ($j = 1, \dots, p$) $p_j = h^t \cdot u_j \geq 1$.

До к а з а т е л ь с т в о. Конечно, из 2) следует 1).

Покажем, что из 1) следует 2).

Пусть для неотрицательных действительных чисел $\alpha_1, \dots, \alpha_p$, из которых не все равны нулю, выполнено равенство

$$\alpha_1 u_1 + \dots + \alpha_p u_p = 0.$$

Без ограничения общности можно считать, что $\alpha_p > 0$.

Равенство

$$(\alpha_1/\alpha_p) \cdot u_1 + \dots + (\alpha_{p-1}/\alpha_p) \cdot u_{p-1} = -u_p$$

означает, что целочисленная линейная система

$$x_1 \cdot u_1 + \dots + x_{p-1} \cdot u_{p-1} = -u_p$$

имеет решение в неотрицательных действительных числах. Тогда в силу двух предыдущих лемм у этой системы имеется базисное решение в неотрицательных рациональных числах $\gamma_1, \dots, \gamma_{p-1}$.

В базисном решении свободные переменные равны нулю, а базисные переменные определяются равенством

$$[\gamma_{j_1}, \dots, \gamma_{j_m}] = B^{-1}(-u_p).$$

Рассмотрим матрицу $B = \|b_{ts}\|$. Напомним, что $B^{-1} = 1/\det(B) \cdot \|B_{st}\|$.

$$\gamma_{j_i} = -\sum_{i=1}^m B_{ti} u_{pi} / \det(B).$$

Так как B_{st} — это минор порядка $m-1$ матрицы B , то

$$|B_{st}| \leq (m-1)! \cdot E^{m-1}.$$

Поэтому

$$|\sum_{i=1}^m B_{ti} u_{pi}| \leq m! \cdot E^m \leq (mE)^m.$$

Кроме того

$$|\det(B)| \leq m! \cdot E^m \leq (mE)^m.$$

Полагаем $\gamma_{jt} = a_{jt}/|\det(B)|$. Так как $\gamma_{jt} \geq 0$, то и $a_{jt} \geq 0$. Причем $a_{jt}, |\det(B)| \leq (mE)^m$. Для свободных переменных x_j полагаем $\gamma_j = a_j/|\det(B)|$, где $a_j = 0$.

Равенство

$$\gamma_1 \cdot u_1 + \dots + \gamma_{p-1} \cdot u_{p-1} = -u_p$$

дает требуемое равенство

$$a_1 \cdot u_1 + \dots + a_{p-1} \cdot u_{p-1} + |\det(B)| \cdot u_p = 0.$$

Покажем, что из 2) (или из 1) следует 3).

Пусть существуют такие неотрицательные числа $\alpha_1, \dots, \alpha_p$, что выполнено равенство

$$\alpha_1 u_1 + \dots + \alpha_p u_p = 0.$$

Предположим, что существует вектор-столбец h высоты m с действительными компонентами такой, что при любом j ($j = 1, \dots, k$): $p_j = h^t \cdot v_j > 0$. Тогда получаем неравенства

$$0 = h^t \cdot \sum_{i=1}^k \alpha_i \cdot u_i = \sum_{i=1}^k \alpha_i \cdot (h^t \cdot u_i) > 0.$$

Полученное противоречие показывает, что из 2) (или из 1) следует 3).

То же самое рассуждение показывает, что из 2) следует 4). Конечно, из 3) следует 4).

Остается показать, что из 4) следует 1) или 2).

Предположим, что выполнено 4), т.е. не существует вектора-столбца h высоты m с целыми компонентами, не превосходящими по модулю числа $(mE)^{m+1}$, такого, что при любом j ($j = 1, \dots, p$) $p_j = h^t \cdot u_j \geq 1$.

Покажем, что несовместна система линейных неравенств

$$\bigwedge_{j=1}^p h^t \cdot u_j \geq 1.$$

Эта система, имеющая вид,

$$\bigwedge_{j=1}^p (h_1, \dots, h_m) \cdot [u_{1j}, \dots, u_{mj}] \geq 1,$$

совместна тогда и только тогда, когда имеет неотрицательное решение система линейных уравнений

$$\bigwedge_{j=1}^p (h_1^+ - h_1^-, \dots, h_m^+ - h_m^-) \cdot [u_{1j}, \dots, u_{mj}] - y_j = 1.$$

В силу выше доказанных лемм у последней системы имеется неотрицательное решение тогда и только тогда, когда у нее имеется неотрицательное базисное (опорное) решение. Базисное решение может быть найдено следующим образом:

выделяем подходящие базисные столбцы и соответствующие им базисные переменные,

небазисные (свободные) переменные обнуляем,

из полученной системы по правилу Крамера находим значения базисных переменных.

Поэтому у последней системы имеется неотрицательное рациональное решение

$$h_1^+ = a_1^+/d, h_1^- = a_1^-/d, \dots, h_m^+ = a_m^+/d, h_m^- = a_m^-/d, \\ y_1 = b_1^-/d, \dots, y_p = b_p^-/d,$$

в котором числители неотрицательны и не превосходят числа $m!E^m$, а общий знаменатель d удовлетворяет неравенствам $1 \leq d \leq m!E^m$. Тогда при любом j ($j = 1, \dots, m$) выполняется неравенство

$$|a_j^+ - a_j^-| \leq m!E^m.$$

Кроме того выполнены и все неравенства

$$\bigwedge_{j=1}^p (a_1^+ - a_1^-, \dots, a_m^+ - a_m^-) \cdot [u_{1j}, \dots, u_{mj}] \geq d \geq 1,$$

что противоречит предположению о выполненности 4).

Значит, несовместна и система линейных неравенств $\bigwedge_{j=1}^p u_j^t \cdot h \leq -1$.

В соответствии с критерием несовместности систем линейных неравенств система линейных уравнений

$$\begin{pmatrix} u_{11} & \dots & u_{1p} \\ \dots & \dots & \dots \\ u_{m1} & \dots & u_{mp} \\ -1 & \dots & -1 \end{pmatrix} \cdot y = \begin{pmatrix} 0 \\ \dots \\ 0 \\ -1 \end{pmatrix}$$

имеет неотрицательное решение $(\alpha_1, \dots, \alpha_p)$.

Значит, выполняются равенства

$$\alpha_1 \cdot u_1 + \dots + \alpha_p \cdot u_p = 0, \quad \alpha_1 + \dots + \alpha_p = 1.$$

Поэтому 1) выполнено. □

Теорема 2.6. Если система линейных уравнений

$$\begin{array}{ccccccc} a_{11}x_1 & + & \dots & + & a_{1n}x_n & = & b_1 \\ & & \dots & & \dots & & \\ a_{m1}x_1 & + & \dots & + & a_{mn}x_n & = & b_m \end{array}$$

с целыми коэффициентами имеет неотрицательное целочисленное решение, то она имеет и неотрицательное целочисленное решение, все компоненты которого не превосходят числа

$$M = n \cdot (mC)^{2m+3} \cdot (D + 1),$$

где

$$C = \max_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} |a_{ij}|, \quad D = \max_{1 \leq i \leq m} |b_i|.$$

Д о к а з а т е л ь с т в о. Пусть $(x_1^{(0)}, \dots, x_n^{(0)})$ – неотрицательное целочисленное решение этой системы с наименьшей суммой компонент.

Если все компоненты этого решения не превосходят $C_1 = (mC)^{m+1}$, то доказываемое утверждение справедливо.

Предположим, что некоторые компоненты этого решения больше C_1 . Без ограничения общности можем считать, что первые p компонент больше C_1 , а остальные не превосходят этого числа. Обозначим столбцы матрицы A через u_1, \dots, u_n .

Возможны два случая.

1) Существуют неотрицательные целые числа $\alpha_1, \dots, \alpha_p$, не превосходящие числа $C_1 = (mC)^{m+1}$ и не все равные нулю, для которых выполнено равенство

$$\alpha_1 u_1 + \dots + \alpha_p u_p = 0.$$

Тогда $(x_1^{(0)} - \alpha_1, \dots, x_p^{(0)} - \alpha_p, x_{p+1}^{(0)}, \dots, x_n^{(0)})$ – неотрицательное целочисленное решение этой системы с меньшей суммой компонент, чем у решения $(x_1^{(0)}, \dots, x_n^{(0)})$, что противоречит сделанному предположению.

2) Не существуют неотрицательных целых чисел $\alpha_1, \dots, \alpha_p$, не превосходящих числа $(mC)^{m+1}$ и не всех равных нулю, для которых выполнено равенство

$$\alpha_1 u_1 + \dots + \alpha_p u_p = 0.$$

Тогда в силу предыдущей леммы существует вектор-столбец h высоты m с целыми компонентами, не превосходящими по модулю числа $(mC)^{m+1}$, такой, что при любом j ($j = 1, \dots, p$) $h^t \cdot u_j \geq 1$.

Из равенства $Ax^{(0)} = b$ получаем равенство

$$\sum_{j=1}^p h^t \cdot u_j \cdot x_j^{(0)} = h^t \cdot b - \sum_{j=p+1}^n h^t \cdot u_j \cdot x_j^{(0)}.$$

Из неравенств $h^t \cdot u_j \geq 1$ и неотрицательности $x_j^{(0)}$ ($j = 1, \dots, p$) следует неравенство

$$\sum_{j=1}^p x_j^{(0)} \leq \sum_{j=1}^p h^t \cdot u_j \cdot x_j^{(0)},$$

а значит, и неравенства

$$\sum_{j=1}^p x_j^{(0)} \leq h^t \cdot b - \sum_{j=p+1}^n h^t \cdot u_j \cdot x_j^{(0)} \leq$$

$$mC_1D + (n-p)mCC_1^2 \leq M.$$

Значит, каждая компонента решения $(x_1^{(0)}, \dots, x_n^{(0)})$ не превосходит числа M .
□

Следствие. Если система линейных неравенств

$$\begin{aligned} a_{11}x_1 + \dots + a_{1n}x_n &\leq b_1 \\ \dots &\dots \\ a_{m1}x_1 + \dots + a_{mn}x_n &\leq b_m \end{aligned}$$

с целыми коэффициентами имеет неотрицательное целочисленное решение, то она имеет и неотрицательное целочисленное решение, все компоненты которого не превосходят числа

$$M_1 = (n+m) \cdot (mC)^{2m+3} \cdot (D+1).$$

Д о к а з а т е л ь с т в о. Доказательство получается сведением вопроса о разрешимости в целых неотрицательных числах x_1, \dots, x_n системы линейных неравенств $Ax \leq b$ к вопросу о разрешимости в целых неотрицательных числах $x_1, \dots, x_n, y_1, \dots, y_m$ системы линейных уравнений $(A, E_m)[x, y] = b$. □

В качестве *размера входа* L в задаче о разрешимости в целых неотрицательных числах системы линейных уравнений $Ax = b$ или неравенств $Ax \leq b$ по ряду естественных причин часто берется число

$$L = mn + \log |P|,$$

где P – произведение всех ненулевых элементов матрицы A и вектора b ($\log |P|$ дает общую длину записи ненулевых элементов матрицы A и вектора b , а mn ограничивает сверху число нулевых элементов матрицы A).

Получаем равенства

$$\log M = \log n + (2m+3) \cdot (\log m + \log C) + \log(D+1) = O(L^2),$$

которые показывают, что задача о разрешимости в целых неотрицательных числах системы линейных уравнений $Ax \leq b$ или системы линейных неравенств $Ax \leq b$ лежит в классе \mathcal{NP} .

Заметим, что задача о разрешимости в целых неотрицательных числах системы линейных уравнений $Ax = b$ и задача о разрешимости в целых неотрицательных числах системы линейных неравенств $Ax \leq b$ полиномиально эквивалентны, т.е. каждая из них полиномиально сводима к другой.

Вопрос о разрешимости в целых неотрицательных числах x_1, \dots, x_n системы линейных неравенств $Ax \leq b$ полиномиально сводится к вопросу о разрешимости в целых неотрицательных числах $x_1, \dots, x_n, y_1, \dots, y_m$ системы линейных уравнений $(A, E_m)[x, y] = b$.

В свою очередь вопрос о разрешимости в целых неотрицательных числах x_1, \dots, x_n системы линейных уравнений $Ax = b$ полиномиально сводится к вопросу о разрешимости в целых неотрицательных числах x_1, \dots, x_n системы линейных неравенств $Ax \leq b \ \& \ (-Ax) \leq (-b)$.

Покажем, что вопрос о разрешимости в целых неотрицательных числах систем линейных уравнений полиномиально эквивалентен вопросу о разрешимости систем линейных уравнений в числах 0, 1.

Система линейных уравнений $Ax = b$ с неизвестными x_1, \dots, x_n имеет решение в числах 0, 1 тогда и только тогда, когда имеет неотрицательное решение система линейных уравнений

$$Ax = b \ \& \ x_1 + y_1 = 1 \ \& \ \dots \ \& \ x_n + y_n = 1$$

с неизвестными $x_1, \dots, x_n, y_1, \dots, y_n$. Таким образом, вопрос о разрешимости в числах 0, 1 систем линейных уравнений полиномиально сводится к вопросу о разрешимости систем линейных уравнений в неотрицательных целых числах.

Покажем, что вопрос о разрешимости в неотрицательных целых числах систем линейных уравнений полиномиально сводится к вопросу о разрешимости систем линейных уравнений в числах 0, 1.

Рассмотрим произвольную систему линейных уравнений

$$\begin{array}{ccccccc} a_{11}x_1 + \dots & + & a_{1n}x_n & = & b_1 \\ & \dots & & & \dots \\ a_{m1}x_1 + \dots & + & a_{mn}x_n & = & b_m \end{array}$$

с целыми коэффициентами. По теореме 2.6, она имеет неотрицательное целочисленное решение тогда и только тогда, когда она имеет неотрицательное целочисленное решение, все компоненты которого не превосходят числа

$$M = n \cdot (mC)^{2m+3} \cdot (D + 1),$$

где

$$C = \max_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}} |a_{ij}|, \quad D = \max_{1 \leq i \leq m} |b_i|.$$

Пусть $s = [\log_2 M] + 1$, где $[a]$ – целая часть числа a . Каждое целое число из отрезка $[a, M]$ можно представить в виде $\sum_{i=0}^s \alpha_i 2^i$, где α_i – это 0 или 1. Это дает возможность заменить каждую переменную x_j набором переменных $x_{j,0}, x_{j,1}, \dots, x_{j,s}$, воспользовавшись равенством

$$x_j = \sum_{i=0}^s x_{j,i} \cdot 2^i.$$

Поэтому система линейных уравнений $Ax = b$ имеет решение в неотрицательных целых числах тогда и только тогда, когда имеет решение в числах 0, 1 следующая система линейных уравнений

$$A \cdot \left[\sum_{i=0}^s x_{1,i} \cdot 2^i, \dots, \sum_{i=0}^s x_{n,i} \cdot 2^i \right] = b.$$

§3. \mathcal{NP} -полнота проблемы разрешимости для уравнений с нетривиальной правой частью в свободной полугруппе

Обозначим через Π_2 свободную полугруппу с пустым словом в качестве нейтрального элемента (свободный моноид) ранга 2 со свободными образующими a и b , т.е. это множество всех слов в двубуквенном алфавите $\{a, b\}$ с операцией умножения (конкатенации) слов.

Системой уравнений с неизвестными x_1, \dots, x_n в свободной полугруппе Π_2 называется выражение вида

$$\bigotimes_{i=1}^k w_i(x_1, \dots, x_n, a, b) = u_i(x_1, \dots, x_n, a, b), \quad (1)$$

где $w_i(x_1, \dots, x_n, a, b)$ и $u_i(x_1, \dots, x_n, a, b)$ – слова в алфавите

$$\{x_1, x_2, \dots, x_n, a, b\}.$$

Набор $\langle g_1, \dots, g_n \rangle$ элементов полугруппы Π_2 называется *решением* системы (1), если при любом i ($i = 1, \dots, k$) в полугруппе Π_2 выполняется равенство

$$w_i(g_1, \dots, g_n, a, b) = u_i(g_1, \dots, g_n, a, b).$$

Две системы уравнений с одними и теми же неизвестными называются *эквивалентными*, если множества их решений совпадают.

Система уравнений

$$\bigotimes_{i=1}^k w_i = u_i$$

равносильна одному уравнению

$$w_1 a_1 w_2 a_1 \dots a_1 w_k w_1 a_2 w_2 a_2 \dots a_2 w_k = u_1 a_1 u_2 a_1 \dots a_1 u_k u_1 a_2 u_2 a_2 \dots a_2 u_k.$$

В конце 50-х годов прошлого века А.А. Марков предложил использовать системы уравнений в свободной полугруппе Π_2 в качестве подхода к отрицательному решению 10-ой проблемы Д. Гильберта. Системы уравнений в свободных полугруппах также называются системами уравнений в словах. Для уравнений в свободных полугруппах традиционно рассматриваются две основные задачи:

проблема существования решения и проблема описания множества всех решений.

Первые результаты в исследовании систем уравнений в словах были получены А.А. Марковым и Ю.И. Хмелевским [86] в конце 1960-х годов.

В 1976 году Г.С. Маканин получил в теории уравнений в словах фундаментальный результат, который был опубликован в 1977 году в работе [33], — он построил алгоритм, позволяющий по любой системе уравнений в свободной полугруппе Π_2 определить, имеет ли она решение.

В настоящее время не известно нетривиальных нижних оценок сложности задачи разрешимости уравнений в свободных полугруппах. Поэтому представляет определенный интерес нахождение "достаточно простых и естественных" \mathcal{NP} -полных проблем для уравнений в свободных полугруппах.

В уже цитировавшейся ранне монографии М. Гэри и Д. Джонсона "Вычислительные машины и труднорешаемые задачи" в список \mathcal{NP} -полных проблем включена следующая проблема:

Подстановка регулярного выражения.

УСЛОВИЕ. Заданы два конечных алфавита $\Sigma = \{a_1, \dots, a_n\}$ и $X = \{x_1, \dots, x_m\}$, регулярное выражение E в алфавите $\Sigma \cup X$, регулярные выражения E_1, \dots, E_n в алфавите Σ и слово g в этом же алфавите.

ВОПРОС. Существуют ли в языках $L(E), L(E_1), \dots, L(E_n)$ такие слова w, v_1, \dots, v_n , что если в слове w заменить x_1 на v_1, \dots, x_n на v_n , то получим слово g ?

Покажем, что эта проблема остается \mathcal{NP} -полной, если в качестве выражения E взять выражение, определяющее язык $L(E)$, состоящий из одного слова w , а в качестве выражений E_1, \dots, E_n взять такие выражения, что определяемые ими языки $L(E_1), \dots, L(E_n)$ являются универсальным языком Σ^* , т.е. на слова v_1, \dots, v_n не накладывается никаких ограничений.

Это приводит нас к следующей задаче:

" по уравнению вида

$$w(x_1, \dots, x_n) = g(a, b) \quad (*)$$

определить, имеет ли оно решение в Π_2 ".

Слово $w(x_1, \dots, x_n)$ является словом в алфавите неизвестных, т.е. оно не содержит образующих a и b , а слово $g(a, b)$ является словом в алфавите образующих, т.е. оно не содержит неизвестных x_1, \dots, x_n .

Вопрос о разрешимости в свободной группе уравнений вида (*) рассматривался в работах Р. Линдона и П. Шуппа и получил название *the substitution problem for free groups*.

Естественно считать размерностью задачи разрешимости уравнений вида (*) число

$$L \Rightarrow |w(x_1, \dots, x_n)| + |g(a, b)|,$$

где $|A|$ — длина слова A .

Целью этого параграфа является доказательство \mathcal{NP} -полноты задачи разрешимости уравнений вида (*) в свободной полугруппе Π_2 .

Предварительно докажем одну простую, но важную для дальнейшего лемму.

Лемма 3.1. *Если w, y, g – слова полугруппы Π_2 , причем g – непустое слово и b^2 не является подсловом слова g^2 , то имеет место эквивалентность*

$$w^2 y^2 = g^2 b^2 \iff w = g \ \& \ y = b.$$

Д о к а з а т е л ь с т в о. Если $w = g$ и $y = b$, то

$$w^2 y^2 = g^2 b^2.$$

Обратно пусть $w^2 y^2 = g^2 b^2$.

Слово y не может быть пустым, так как в противном случае мы имели бы $w^2 = g^2 b^2$, $|w| = |g| + |b| \geq 2$, поэтому для некоторого слова u выполнялось бы равенство $w = ub^2$, из которого мы могли бы получить равенство $g^2 = ub^2 u$, что противоречит условию леммы.

Предположение о том, что длина y не меньше 2 также ведет к противоречию, так как неравенство $|y| \geq 2$ влечет существование такого слова u , что $y = ub^2$, а это в свою очередь приводит к равенству $g^2 = w^2 ub^2 u$, которое противоречит условию леммы.

Значит, $|y| = 1$, поэтому $y = b$, $w = g$. □

Теорема 3.1. *Проблема разрешимости уравнений вида (*) для свободной полугруппы Π_2 является \mathcal{NP} -полной.*

Д о к а з а т е л ь с т в о. Если x_1^0, \dots, x_n^0 – решение уравнения (*), то

$$|w(x_1^0, \dots, x_n^0)| = |g(a, b)| \leq L$$

и при любом i ($1 \leq i \leq n$)

$$|x_i^0| \leq |g(a, b)| \leq L,$$

более того

$$\sum_{i=1}^n |x_i^0| \leq L,$$

поэтому за полиномиальное от L время мы можем убедиться в справедливости равенства

$$w(x_1^0, \dots, x_n^0) = g(a, b),$$

значит, рассматриваемая задача лежит в классе \mathcal{NP} .

Для доказательства \mathcal{NP} -трудности этой задачи покажем, что к ней полиномиально сводима задача 3-выполнимости (3-ВЫП-задача), являющаяся, как показано выше, \mathcal{NP} -полной задачей.

Напомним, что "3-ВЫП-задача" состоит в определении по формуле логики высказываний, находящейся в конъюнктивной нормальной форме, содержащей в каждом дизъюнкте ровно 3 литерала, является ли эта формула выполнимой.

Пусть формула Φ логики высказываний имеет вид

$$\big\&_{i=1}^n \bigvee_{j \in A_i} X_j^{\varepsilon_{ij}},$$

где $\varepsilon_{ij} \in \{1, -1\}$, причем

$$X_j^1 \Rightarrow X_j, \quad X_j^{-1} \Rightarrow \overline{X_j}.$$

Кроме того при любом i ($1 \leq i \leq m$) множество A_i состоит из трех элементов.

По формуле Φ построим систему Φ^* уравнений и неравенств

$$\big\&_{i=1}^m \sum_{j \in A_i} x_j^{\varepsilon_{ij}} \geq 1 \quad \& \quad \big\&_{j=1}^n x_j + x_j^{-1} = 1$$

с $2n$ неизвестными $x_1, x_1^{-1}, \dots, x_n, x_n^{-1}$.

Заметим, что каждая сумма

$$\sum_{j \in A_i} x_j^{\varepsilon_{ij}}$$

содержит ровно три слагаемых.

Нетрудно понять, что формула Φ выполнима тогда и только тогда, когда система Φ^* имеет решение в числах 0, 1 (значению "истинно" переменной X_j соответствует значение 1 переменной x_j , а значению "ложно" – значение 0).

Неравенство

$$\sum_{j \in A_i} x_j^{\varepsilon_{ij}} \geq 1$$

равносильно неравенству

$$3 - \sum_{j \in A_i} x_j^{\varepsilon_{ij}} \leq 2.$$

В свою очередь последнее неравенство равносильно неравенству

$$\sum_{j \in A_i} (1 - x_j^{\varepsilon_{ij}}) \leq 2.$$

Поэтому система Φ^* имеет решение в числах 0, 1 тогда и только тогда, когда имеет решение в числах 0, 1 следующая система Φ^{**} :

$$\big\&_{i=1}^m \sum_{j \in A_i} x_j^{-\varepsilon_{ij}} \leq 2 \quad \& \quad \big\&_{j=1}^n x_j + x_j^{-1} = 1.$$

По системе Φ^{**} построим систему уравнений Ψ в полугруппе Π_2

$$\big\&_{i=1}^m \prod_{j \in A_i} x_j^{-\varepsilon_{ij}} \cdot y_i = a^2 \quad \& \quad \big\&_{j=1}^n x_j x_j^{-1} = a.$$

Нетрудно понять, что формула Φ выполнима тогда и только тогда, когда система Ψ имеет решение в Π_2 .

Заметим, что система Ψ равносильна в Π_2 уравнению

$$x_1 x_1^{-1} b x_2 x_2^{-1} b \dots b x_n x_n^{-1} b \prod_{j \in A_1} x_j^{-\varepsilon_{1j}} y_1 b \dots b \prod_{j \in A_m} x_j^{-\varepsilon_{mj}} y_m = (ab)^n (a^2 b)^{m-1} a^2,$$

где слева и справа буква b имеет по $n + m - 1$ вхождений.

Обозначим левую часть последнего уравнения через

$$w(x_1, \dots, x_n, b),$$

а правую – через $g(a, b)$.

Уравнение

$$w(x_1, \dots, x_n, b) = g(a, b)$$

имеет решение в Π_2 тогда и только тогда, когда в Π_2 разрешима система уравнений

$$w(x_1, \dots, x_n, y) = g(a, b) \ \& \ y = b,$$

а последняя система, в свою очередь, в силу леммы 3.1 равносильна уравнению

$$w^2(x_1, \dots, x_n, y) y^2 = g^2(a, b) b^2.$$

Итак, **проблема 3-выполнимости для логики высказываний** полиномиально сведена к проблеме разрешимости в Π_2 уравнений вида (*).

Это завершает доказательство теоремы. \square

Замечание 1. Теорема справедлива и для любой свободной полугруппы Π_n с n образующими при $n \geq 2$.

Замечание 2. Естественно рассмотреть и для полугруппы $\Pi_1 = \langle a \rangle$ вопрос о разрешимости уравнений вида

$$w(x_1, \dots, x_n) = g(a).$$

Легко понять, что вопрос о разрешимости в Π_1 уравнения вида

$$w(x_1, \dots, x_n, a) = g(a)$$

полиномиально сводится к решению в неотрицательных целых числах уравнения вида

$$c_1 x_1 + \dots + c_n x_n = b,$$

где c_1, \dots, c_n, b – неотрицательные целые числа, но *размером* последней задачи следует считать число

$$L = \sum_{i=1}^n c_i + b,$$

а не число

$$L_1 = \sum_{i=1}^n [\log c_i] + [b],$$

как это обычно делается при исследовании вопроса о разрешимости в натуральных или целых числах линейных уравнений и их систем. При этом естественно считать, что при любом i : $0 < c_i \leq b$. В [64] доказано, что при этих предположениях вопрос о разрешимости в неотрицательных целых числах уравнения вида

$$c_1 x_1 + \dots + c_n x_n = b$$

можно решить за время $O(nb)$. Значит вопрос о существовании в Π_1 решения у уравнения

$$w(x_1, \dots, x_n, a) = g(a)$$

решается за *полиномиальное время от*

$$|w(x_1, \dots, x_n, a)| + |g(a)|.$$

В связи с этим интересно отметить, что задача разрешимости в Π_1 систем вида

$$\bigotimes_{i=1}^m w_i(x_1, \dots, x_n) = g_i(a)$$

также является \mathcal{NP} -полной задачей.

Доказательство этого факта сразу следует из доказательства предыдущей теоремы, если вспомнить, что *формула*

$$\bigotimes_{i=1}^n \bigvee_{j \in A_i} X_j^{\varepsilon_{ij}}$$

выполнима тогда и только тогда, когда система

$$\bigotimes_{i=1}^m \prod_{j \in A_i} x_j^{-\varepsilon_{ij}} \cdot y_i = a^2 \quad \& \quad \bigotimes_{j=1}^n x_j x_j^{-1} = a$$

имеет решение в Π_1 .

Замечание 3. При любом фиксированном n задача разрешимости в Π_2 уравнений вида

$$w(x_1, \dots, x_n, a, b) = g(a, b) \quad (**)$$

лежит в классе \mathcal{P} . В частности, это верно для уравнения вида (*).

В самом деле, если x_1^0, \dots, x_n^0 – решение уравнения (**), то при любом i ($1 \leq i \leq n$) x_i^0 – *подслово слова* $g(a, b)$.

Если

$$L = |w(x_1, \dots, x_n, a, b)| + |g(a, b)|,$$

то $|g(a, b)| \leq L$, поэтому число подслов слова $g(a, b)$ меньше L^2 , значит, существует менее чем L^{2n} наборов слов g_1, \dots, g_n таких, что при любом i ($1 \leq i \leq n$) g_i – подслово слова $g(a, b)$.

Так как

$$w(x_1^0, \dots, x_n^0) = g(a, b),$$

то

$$|w(x_1^0, \dots, x_n^0)| \leq L \quad \text{и} \quad \sum_{i=1}^n |x_i^0| \leq L,$$

поэтому в качестве претендентов на решение уравнения $(**)$ следует проверять лишь такие наборы g_1, \dots, g_n подслов слова $g(a, b)$, для которых

$$\sum_{i=1}^n |g_i| \leq L \quad \text{и} \quad |w(g_1, \dots, g_n)| \leq L.$$

Эти рассуждения и дают полиномиальный при фиксированном n алгоритм для выяснения, имеет ли уравнение $(**)$ решение в Π_2 , более того, этот же алгоритм позволяет и найти все решения уравнения $(**)$ за полиномиальное время (при фиксированном n).

§4. \mathcal{NP} -полные проблемы в теории графов

Параграф посвящен рассмотрению некоторых \mathcal{NP} -полных проблем из теории графов. При отборе материала мы руководствовались принципом – продемонстрировать, как путем полиномиальной сводимости, отправляясь от \mathcal{NP} -трудной проблемы, можно устанавливать \mathcal{NP} -трудность все новых и новых проблем аналогично тому, как в классической теории алгоритмов, отправляясь от задачи, алгоритмическая неразрешимость которой уже установлена, путем m -сведения можно устанавливать алгоритмическую неразрешимость все новых и новых задач. Конечно, мы ни в коей мере не претендуем на какую-либо полноту охвата материала из этой обширнейшей области. При изложении доказательств мы опирались на источники, указанные в списке литературы в конце пособия.

Через $G = (V, E)$ будем обозначать неориентированный граф с множеством вершин V и множеством ребер E , а через $\vec{G} = (\vec{V}, \vec{E})$ – ориентированный граф с множеством вершин \vec{V} и множеством дуг \vec{E} .

Клика. Требуется по произвольному графу G и натуральному числу k определить, содержит ли граф G полный подграф на k вершинах.

Определение 4.1. Подмножество V_1 множества вершин V графа $G = (V, E)$ называется **независимым**, если никакие две вершины этого подмножества V_1 не связаны ребром.

Независимое множество вершин. Требуется по произвольному графу G и натуральному числу k определить, имеет ли граф G независимое множество из k вершин.

Изоморфность подграфу. Требуется по двум заданным графам G и G_1 определить, изоморфен ли граф G_1 некоторому подграфу графа G .

Определение 4.2. Подмножество V_1 множества вершин V графа $G = (V, E)$ образует **вершинное покрытие** этого графа, если для каждого его ребра $e \in E$ одна из инцидентных ему вершин принадлежит подмножеству V_1 .

Вершинное покрытие. Требуется по произвольному графу G и натуральному числу k определить, имеет ли граф G вершинное покрытие с k вершинами.

Гамильтонов цикл. Требуется по произвольному графу G определить, содержит ли граф G цикл, проходящий через каждую его вершину ровно один раз.

Определение 4.3. Граф G называется **k -раскрашиваемым**, если его вершины могут быть помечены числами от 1 до k (раскрашены в k цветов) таким образом, чтобы смежным вершинам были приписаны различные числа (смежные вершины раскрашены в разные цвета). Наименьшее число k такое, что граф G является k -раскрашиваемым, называется **хроматическим числом** этого графа.

Раскрашиваемость. Требуется по произвольному графу G и натуральному числу k определить, является ли граф G k -раскрашиваемым.

Теорема 4.1. Проблема **Клика** является \mathcal{NP} -полной.

Доказательство. Принадлежность проблемы **Клика** классу \mathcal{NP} очевидна. Покажем, что проблема **Выполнимость** полиномиально сводима к проблеме **Клика**.

Рассмотрим произвольную формулу F в КНФ

$$F = A_1 \& \dots \& A_m.$$

Построим по F граф $G(F)$, вершинами которого будут пары $\langle x_i^\varepsilon, j \rangle$, где литерал x_i^ε входит в элементарную дизъюнкцию A_j (x_i — это переменная, $\varepsilon \in \{-1, 1\}$, x_i^{-1} — это $\neg x_i$).

Вершины $\langle x_i^\varepsilon, j \rangle$ и $\langle x_t^\delta, s \rangle$ соединяются ребром тогда и только тогда, когда $j \neq s$ и литерал x_i^ε не является отрицанием литерала x_t^δ , т.е. $x_i^{-\varepsilon} \neq x_t^\delta$.

Покажем, что в графе $G(F)$ есть m -клика тогда и только тогда, когда формула F выполнима.

Предположим, что формула F выполнима. Рассмотрим произвольный выполняющий набор $\alpha_1, \dots, \alpha_n$. В каждой элементарной дизъюнкции A_j найдется литерал $x_i^{\varepsilon_{i,j}}$, принимающий значение 1. Тогда $\varepsilon_{i,j} = \alpha_i$. Возьмем по

одному такому литералу в каждой элементарной конъюнкции. Из построения графа сразу следует, что соответствующие m вершин в графе $G(F)$ образуют m -клику.

Предположим, что в графе $G(F)$ имеется m -клика $\langle x_{i_1}^{\varepsilon_1}, j_1 \rangle, \dots, \langle x_{i_m}^{\varepsilon_m}, j_m \rangle$.

Полагая $\alpha_{i_1} = \varepsilon_1, \dots, \alpha_{i_m} = \varepsilon_m$ и произвольным образом выбирая значения остальных переменных, получим для формулы F выполняющий набор $\alpha_1, \dots, \alpha_n$. \square

Теорема 4.2. *Проблема **Изоморфность подграфу** является \mathcal{NP} -полной.*

Доказательство. К проблеме **Изоморфность подграфу** полиномиально сводима проблема **Клика**, которая может быть переформулирована как проблема изоморфности полного графа C_k на k вершинах некоторому подграфу графа G . \square

Отметим, что вопрос о сложности следующей достаточно близкой задачи в настоящее время остается открытым.

Изоморфизм графов.

По двум графам $G_1 = (V_1, E_1)$ и $G_2 = (V_2, E_2)$ требуется определить, изоморфны ли они.

Теорема 4.3. *Проблема **Вершинное покрытие** является \mathcal{NP} -полной.*

Доказательство. Принадлежность проблемы **Вершинное покрытие** классу \mathcal{NP} очевидна.

Покажем, что проблема **Клика** полиномиально сводима к проблеме **Вершинное покрытие**. По произвольному неориентированному графу $G = (V, E)$ построим неориентированный граф $\overline{G} = (V, \overline{E})$, являющийся дополнением графа $G = (V, E)$ до полного графа на множестве вершин V , т.е. если v и u – две различные вершины, то они в графе $\overline{G} = (V, \overline{E})$ соединяются ребром тогда и только тогда, когда они в графе $G = (V, E)$ не были соединены ребром.

Покажем, что подмножество V_1 множества вершин V образует клику в графе $G = (V, E)$ тогда и только тогда, когда подмножество $V \setminus V_1$ множества вершин V образует в графе $\overline{G} = (V, \overline{E})$ вершинное покрытие.

Если подмножество V_1 множества вершин V образует клику в графе $G = (V, E)$, то никакие две различные вершины этого подмножества в графе $\overline{G} = (V, \overline{E})$ не соединены ребром, значит, для любого ребра графа $\overline{G} = (V, \overline{E})$ одна из его концевых вершин не принадлежит подмножеству V_1 , т.е. принадлежит подмножеству $V \setminus V_1$. Поэтому подмножество $V \setminus V_1$ множества вершин V образует в графе $\overline{G} = (V, \overline{E})$ вершинное покрытие.

Для доказательства обратного утверждения, предположим, что подмножество $V \setminus V_1$ множества вершин V образует в графе $\overline{G} = (V, \overline{E})$ вершинное покрытие. Покажем, что подмножество V_1 множества вершин V образует клику в графе $G = (V, E)$. Предположим противное, пусть вершины v и u этого подмножества V_1 в графе $G = (V, E)$ не соединены ребром. Тогда вершины v и u в

графе $\overrightarrow{G} = (V, \overrightarrow{E})$ соединены некоторым ребром e . Так как подмножество $V \setminus V_1$ множества вершин V образует в графе $\overrightarrow{G} = (V, \overrightarrow{E})$ вершинное покрытие, то одна из концевых вершин v или u ребра e должна принадлежать подмножеству $V \setminus V_1$, что противоречит предположению $v, u \in V_1$. \square

Теорема 4.4. *Проблема Гамильтонов цикл для ориентированного графа является \mathcal{NP} -полной.*

Доказательство. Покажем, что проблема **Вершинное покрытие** полиномиально сводима к проблеме **Гамильтонов цикл** для ориентированного графа.

По произвольному неориентированному графу $G = (V, E)$ и натуральному числу k построим за полиномиальное время такой ориентированный граф $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$, что в графе G имеется вершинное покрытие с k вершинами тогда и только тогда, когда в ориентированном графе $\overrightarrow{G} = (\overrightarrow{V}, \overrightarrow{E})$ имеется гамильтонов цикл.

Предположим, что $V = \{v_1, \dots, v_n\}$. Ребро, соединяющее вершины v_i и v_j , обозначаем через e_{ij} . Так как рассматривается неориентированный граф, то $e_{ij} = e_{ji}$. Выберем k новых символов a_1, \dots, a_k .

Для построения множества \overrightarrow{V} вершин графа \overrightarrow{G} сопоставим каждому новому символу a_i одну вершину, которую будем обозначать тем же символом a_i . Каждому ребру e_{ij} графа G с концевыми вершинами v_i и v_j сопоставим четыре вершины графа \overrightarrow{G}

$$[v_i, e_{ij}, 0], [v_i, e_{ij}, 1], [v_j, e_{ij}, 0], [v_j, e_{ij}, 1]$$

и шесть ориентированных ребер

$$\begin{aligned} [v_i, e_{ij}, 0] &\rightarrow [v_i, e_{ij}, 1], & [v_j, e_{ij}, 0] &\rightarrow [v_j, e_{ij}, 1], \\ [v_i, e_{ij}, 0] &\rightarrow [v_j, e_{ij}, 0], & [v_j, e_{ij}, 0] &\rightarrow [v_i, e_{ij}, 0], \\ [v_i, e_{ij}, 1] &\rightarrow [v_j, e_{ij}, 1], & [v_j, e_{ij}, 1] &\rightarrow [v_i, e_{ij}, 1]. \end{aligned}$$

Представим граф G в виде n списков – по одному списку для каждой вершины этого графа. Список для вершины v_i содержит в некотором фиксированном порядке все ребра графа G , которым инцидентна эта вершина.

Приступаем к построению ориентированных ребер графа \overrightarrow{G} .

Из каждой вершины a_t ($t = 1, \dots, k$) проведем ребро в каждую вершину вида $[v_i, e_{is}, 0]$, где e_{is} – первое ребро в списке для вершины v_i .

В каждую вершину a_t ($t = 1, \dots, k$) проведем ребро из каждой вершины вида $[v_i, e_{is}, 1]$, где e_{is} – последнее ребро в списке для вершины v_i .

Из вершины $[v_i, e_{it}, 1]$ в вершину $[v_i, e_{is}, 0]$ проводим ребро, если e_{is} непосредственно следует за ребром e_{it} в списке ребер для вершины v_i .

Эти ребра вместе с ранее построенными ребрами образуют полный набор \overrightarrow{E} ребер графа \overrightarrow{G} .

Остается доказать, что в графе G имеется вершинное покрытие с k вершинами тогда и только тогда, когда в ориентированном графе $\vec{G} = (\vec{V}, \vec{E})$ имеется гамильтонов цикл.

Предположим, что вершины v_1, \dots, v_k образуют в графе G вершинное покрытие.

Для каждого i ($i = 1, \dots, k$) обозначим через e_{i1}, \dots, e_{it_i} список ребер, которым инцидентна вершина v_i .

Рассмотрим цикл

$$\begin{aligned}
 a_1 &\rightarrow [v_1, e_{11}, 0], [v_1, e_{11}, 0] \rightarrow [v_1, e_{11}, 1], \\
 &[v_1, e_{11}, 1] \rightarrow [v_1, e_{12}, 0], [v_1, e_{12}, 0] \rightarrow [v_1, e_{12}, 1], \\
 &\dots \\
 &[v_1, e_{1t_1}, 0] \rightarrow [v_1, e_{1t_1}, 1], [v_1, e_{1t_1}, 1] \rightarrow a_2, \\
 &a_2 \rightarrow [v_2, e_{21}, 0], [v_2, e_{21}, 0] \rightarrow [v_2, e_{21}, 1], \\
 &[v_2, e_{21}, 1] \rightarrow [v_2, e_{22}, 0], [v_2, e_{22}, 0] \rightarrow [v_2, e_{22}, 1], \\
 &\dots \\
 &[v_2, e_{2t_2}, 0] \rightarrow [v_2, e_{2t_2}, 1], [v_2, e_{2t_2}, 1] \rightarrow a_3, \\
 &\dots \\
 &a_k \rightarrow [v_k, e_{k1}, 0], [v_k, e_{k1}, 0] \rightarrow [v_k, e_{k1}, 1], \\
 &[v_k, e_{k1}, 1] \rightarrow [v_k, e_{k2}, 0], [v_k, e_{k2}, 0] \rightarrow [v_k, e_{k2}, 1], \\
 &\dots \\
 &[v_k, e_{kt_k}, 0] \rightarrow [v_k, e_{kt_k}, 1], [v_k, e_{kt_k}, 1] \rightarrow a_1.
 \end{aligned}$$

Построенный цикл не проходит через вершины графа \vec{G} , построенные по спискам ребер для вершин v_j при $j > k$.

Рассмотрим вершину v_j при $j > k$ и ей инцидентное ребро e .

В графе \vec{G} имеются две вершины $[v_j, e, 0]$ и $[v_j, e, 1]$, не входящие в построенный цикл. Так как вершины v_1, \dots, v_k образуют в графе G вершинное покрытие, то найдется такое i ($1 \leq i \leq k$), что вершина v_i инцидентна ребру e . Заменяем в рассматриваемом цикле ребро $[v_i, e, 0] \rightarrow [v_i, e, 1]$ на путь

$$[v_i, e, 0] \rightarrow [v_j, e, 0], [v_j, e, 0] \rightarrow [v_j, e, 1], [v_j, e, 1] \rightarrow [v_i, e, 1].$$

После указанных преобразований получим в графе \vec{G} гамильтонов цикл.

Для доказательства обратного предположим, что в графе \vec{G} имеется гамильтонов цикл.

Этот путь можно разбить на k подпутей, начальные и конечные вершины которых принадлежат множеству

$$\{a_1, a_2, \dots, a_k\}.$$

Обозначим путь указанного вида, ведущий из вершины a_t в вершину a_s через L_{ts} .

Получаем k путей $L_{s_1 s_2}, L_{s_2 s_3}, \dots, L_{s_{k-1} s_k}$, где s_1, \dots, s_k – перестановка чисел $1, \dots, k$. Анализ пути $L_{s_t s_{t+1}}$, ведущего из вершины a_{s_t} в вершину $a_{s_{t+1}}$ и не содержащего в качестве промежуточных вершин a_p , показывает, что из вершины a_{s_t} мы попадаем в вершину вида $[v, e, 0]$, где e – первое ребро в списке ребер инцидентных вершине v , затем можем посещать вершины вида $[v, f, 0]$ и $[v, f, 1]$, где ребро f инцидентно вершине v и вида $[u, f, 0]$ и $[u, f, 1]$, где ребро f инцидентно вершинам v и u . В вершину $a_{s_{t+1}}$ можем попасть лишь по ребру вида $[v, g, 1]$, где g – последнее ребро в списке ребер инцидентных вершине v . Пути, $L_{s_t s_{t+1}}$, ведущему из вершины a_{s_t} в вершину $a_{s_{t+1}}$, сопоставим указанную вершину v , которую обозначим через v_{s_t} .

Покажем, что вершины v_{s_1}, \dots, v_{s_k} образуют в графе G вершинное покрытие.

Рассмотрим произвольное ребро e графа G с концевыми вершинами v и u . Если $[v, e, 0] \in L_{s_t s_{t+1}}$, то либо v , либо u содержится в списке вершин v_{s_1}, \dots, v_{s_k} . \square

Теорема 4.5. *Проблема Гамильтонов цикл для неориентированных графов является \mathcal{NP} -полной.*

Д о к а з а т е л ь с т в о. Покажем, что проблема **Гамильтонов цикл** для ориентированного графа полиномиально сводима к проблеме **Гамильтонов цикл** для неориентированного графа.

По произвольному ориентированному графу $\vec{G} = (\vec{V}, \vec{E})$ построим неориентированный граф $G = (V, E)$ такой, что в ориентированном графе $\vec{G} = (\vec{V}, \vec{E})$ имеется гамильтонов цикл тогда и только тогда, когда гамильтонов цикл есть в неориентированном графе $G = (V, E)$.

Полагаем $V = \vec{V} \times \{0, 1, 2\}$.

Множество E ребер графа G состоит из ребер, соединяющих вершины $(v, 0)$ и $(v, 1)$, $(v, 1)$ и $(v, 2)$. Кроме того, во множество E входит ребро, соединяющее вершины $(v, 2)$ и $(u, 0)$ тогда и только тогда, когда в ориентированном графе $\vec{G} = (\vec{V}, \vec{E})$ было ребро $v \rightarrow u$.

Предположим, что в построенном графе G имеется гамильтонов цикл. Можно считать, что он начинается с вершины вида $(v, 0)$, тогда вторые компоненты вершин этого цикла дают последовательность $0, 1, 2, 0, 1, 2, \dots, 0, 1, 2, 0$.

Тогда последовательность переходов $2, 0, 2, \dots, 0, 2, 0$ дает гамильтонов цикл в ориентированном графе $\vec{G} = (\vec{V}, \vec{E})$.

В свою очередь по гамильтонову циклу в ориентированном графе $\vec{G} = (\vec{V}, \vec{E})$ можно построить цикл в неориентированном графе G , заменив в нем каждое ребро $v \rightarrow u$ на путь

$$(v, 0), (v, 1), (v, 2), (u, 0).$$

\square

Теорема 4.6. *Проблема **Раскрашиваемость** для графов является \mathcal{NP} -полной.*

Доказательство. Покажем, что проблема **3-выполнимость** полиномиально сводима к проблеме **Раскрашиваемость**.

По произвольной формуле F в 3-КНФ от n переменных x_1, \dots, x_n и с k элементарными дизъюнкциями F_1, \dots, F_k за полиномиальное время от $\max(n, k)$ построим неориентированный граф $G = (V, E)$ с $3n + k$ вершинами такой, что *формула F выполнима тогда и только тогда, когда граф $G = (V, E)$ можно раскрасить в $n + 1$ цветов.*

Введем n новых переменных v_1, \dots, v_n . Считаем, что $n \geq 4$.

Определяем множество V вершин графа $G = (V, E)$

$$V = \{x_1, \dots, x_n, \overline{x_1}, \dots, \overline{x_n}, v_1, \dots, v_n, F_1, \dots, F_k\}.$$

Множество ребер E графа G состоит из ребер четырех типов:

- 1) всевозможные ребра вида $\{v_i, v_j\}$ при $i \neq j$,
- 2) всевозможные ребра вида $\{v_i, x_j\}$ и $\{v_i, \overline{x_j}\}$ при $i \neq j$,
- 3) всевозможные ребра вида $\{x_i, \overline{x_i}\}$ при $1 \leq i \leq n$,
- 4) всевозможные ребра вида $\{x_i, F_j\}$ при условии, что литерал x_i не входит в элементарную дизъюнкцию F_j и $\{\overline{x_i}, F_j\}$ при условии, что литерал $\overline{x_i}$ не входит в элементарную дизъюнкцию F_j .

Покажем, что *формула F выполнима тогда и только тогда, когда граф $G = (V, E)$ можно раскрасить в $n + 1$ цветов.*

Предположим, что граф $G = (V, E)$ можно раскрасить в $n + 1$ цветов. Так как вершины v_1, \dots, v_n образуют полный подграф K_n на n вершинах, то для их раскраски необходимо n цветов. Так как вершины x_j и $\overline{x_j}$ смежны и каждая из них смежна с любой из вершин v_i при $i \neq j$, то одна из вершин x_j и $\overline{x_j}$ того же цвета, что и вершина v_j , а другая – нового цвета, который часто называют *специальным* цветом.

Той из переменных x_j и $\overline{x_j}$, которая раскрашена в специальный цвет, присвоим значение 0, а другой – 1. Покажем, что этот набор значений переменных выполняет формулу F . Покажем, что *ни одна из вершин F_j не раскрашена в специальный цвет.* Вершина F_j смежна с $2n - 3$ из $2n$ вершин

$$x_1, \dots, x_n, \overline{x_1}, \dots, \overline{x_n}.$$

Так как по предположению $n \geq 4$, то $2n - 3 > n$, поэтому для любого j найдется такое i , что вершина F_j смежна как с вершиной x_i , так и с вершиной $\overline{x_i}$. Но одна из них специального цвета, поэтому *вершина F_j не раскрашена в специальный цвет.*

Если в F_j входит литерал v такой, что литерал \overline{v} раскрашен в специальный цвет, то вершина F_j не смежна ни с какой вершиной одного цвета с v , поэтому вершину F_j можно раскрасить в тот же цвет, что и v . Иначе необходим новый цвет. Значит, *для раскраски всех F_1, \dots, F_k не потребуется нового цвета*

тогда и только тогда, когда литералам можно так приписать специальный цвет, что каждая дизъюнкция F_j содержит такой литерал v , что литерал \bar{v} раскрашен в специальный цвет. Другими словами, граф $G = (V, E)$ можно раскрасить в $n + 1$ цветов тогда и только тогда, когда переменным можно так присвоить значения 0 и 1, чтобы в каждой дизъюнкции F_j содержался такой литерал v , что литералу \bar{v} присвоено значение 0 (он раскрашен в специальный цвет). Последнее равносильно выполнимости формулы F . \square

Несколько более сложное рассуждение показывает, что даже проблема **Раскрашиваемость графа в 3 цвета** является \mathcal{NP} -полной.

Аналогично проблема **Раскрашиваемость** для графов является \mathcal{NP} -полной при любом фиксированном $k \geq 3$.

В то же время проблема **Раскрашиваемость графа в 2 цвета** полиномиально разрешима:

выбираем произвольную вершину и помечаем ее числом 0, все смежные с ней вершины помечаем числом 1, все смежные с ними помечаем 0 и т.д. Если в ходе раскраски "возникнет конфликт", т.е. некоторую вершину потребуется пометить и 0, и 1, то 2-раскраска невозможна, если окажутся 2-раскрашенными все вершины одной компоненты связности, то переходим к другой компоненте и т.д.

В заключение отметим полиномиальную разрешимость одной проблемы, в определенном смысле близкой по формулировке проблемам, рассмотренным в этом параграфе.

Эйлеров цикл. Требуется по произвольному графу G определить, содержит ли граф G эйлеров цикл, т.е. цикл, проходящий по каждому его ребру ровно один раз.

Существование полиномиального алгоритма следует из следующего утверждения:

связный граф содержит эйлеров цикл тогда и только тогда, когда степени всех его вершин четны.

§5. Алгоритмически неразрешимые проблемы в области защиты информации

В этом параграфе устанавливается алгоритмическая неразрешимость одной проблемы, связанной с защитой информации посредством реализации дискреционной политики безопасности.

Дискреционный контроль и управление доступом.

При описании политик безопасности в настоящее время наибольшее распространение получили **субъектно-объектные модели**. При таком подходе

все вопросы безопасности информации в автоматизированных системах описываются в терминах доступов субъектов к объектам.

Это утверждение рассматривается в качестве *основной аксиомы политики безопасности*.

Объект доступа рассматривается как *пассивная именованная сущность* – составляющая компьютерной системы, используемая для хранения, обработки или получения информации, к которой можно обратиться по имени. Например, в Windows 2000 в число объектов входят файлы, устройства, каналы, почтовые ящики, процессы и т.д. Объект доступа выступает в качестве неделимой единицы информационного ресурса автоматизированной системы. Порядок доступа субъектов к объектам определяется правилами разграничения доступа.

Субъект доступа – это *активная именованная сущность*, которая может запрашивать ресурсы, использовать их для выполнения тех или иных операций и т.д.

Взаимодействие субъектов с объектами, при котором происходит перенос информации между ними, носит название *доступ*.

Доступ рассматривается как операция ознакомления с информацией или ее обработка – копирование, изменение, уничтожение и т.д. Выделяются, в частности, две основные операции, при выполнении которых происходит перенос информации между субъектами и объектами, – это операция чтения и операция записи. Виды доступа различны для разных типов объектов. Например, для файлов в качестве типичных прав доступа выступают права чтения (Read), записи (Write), выполнения (Execute) и владения (Own).

Действующая в системе политика разграничения доступа разрешает или запрещает те или иные доступы объектов к субъектам. На сегодняшний день широкое распространение получил механизм контроля и управления доступом в компьютерных системах, основанный на дискреционной политике безопасности.

Дискреционное управление доступом есть разграничение доступа между поименованными субъектами и поименованными объектами. Для задания правил разграничения доступа используется *матрица доступа*, строки которой помечены субъектами, а столбцы – объектами. На пересечении строки, помеченной *субъектом S* и столбца, помеченного *объектом O*, находятся *права доступа субъекта S к объекту O*.

Харрисон, Рузо и Ульман в 1976 году опубликовали формальное описание математической модели для исследования дискреционной политики безопасности, получившей название HRU-модель.

HRU-модель включает в себя

- 1) счетное множество **O** *объектов*,
- 2) счетное множество **S** *субъектов*, при этом считается, что каждый субъект является и объектом, т.е. множество **S** субъектов есть подмножество множества **O** объектов,
- 3) конечное множество $\mathbf{R} = \{r_1, \dots, r_n\}$ *прав доступа*,
- 4) *матрицу доступа M*, строки которой помечены элементами некоторого конечного подмножества **S(M)** множества **S** субъектов, а столбцы – элементами некоторого конечного подмножества **O(M)** множества **O** объектов, элементами матрицы доступа **M** являются подмножества множества **R** прав

доступа. Через $\mathbf{M}[s, o]$ будет обозначаться подмножество множества \mathbf{R} прав доступа, стоящее в матрице \mathbf{M} на пересечении строки, помеченной субъектом s и столбца, помеченного объектом o .

Множества \mathbf{O} и \mathbf{S} можно считать потенциально счетными, т.е. в каждый момент конечными, но с возможностью неограниченного увеличения числа элементов. Содержательно это можно понимать, например, как возможность неограниченного расширения числа пользователей или создания все новых и новых файлов, программ и т.д. Так как субъекты и объекты – это именованные сущности, то можно было бы фиксировать конечный алфавит Σ и считать, что субъекты и объекты – это просто слова в алфавите Σ , т.е. $\mathbf{O}=\mathbf{S}=\Sigma^*$. Но такой подход, возможно, слишком абстрактен.

Матрица доступа – основной элемент дискреционной политики безопасности. Управление осуществляется путем изменений матрицы доступа, которые определяются следующими шестью примитивными операторами, где через \mathbf{M} и \mathbf{M}' будут обозначаться соответственно текущая матрица доступа и матрица доступа, полученная из \mathbf{M} выполнением примитивного оператора,

1) *enter r into (s, o)* – внесение права доступа r в ячейку матрицы (s, o) ; в результате выполнения этого оператора матрица \mathbf{M} преобразуется в матрицу \mathbf{M}' , строки и столбцы которой помечены так же, как строки и столбцы матрицы \mathbf{M} , т.е. $\mathbf{S}(\mathbf{M}') = \mathbf{S}(\mathbf{M})$ и $\mathbf{O}(\mathbf{M}') = \mathbf{O}(\mathbf{M})$, $\mathbf{M}'[s, o] = \mathbf{M}[s, o] \cup \{r\}$ и $(\forall u \in S(\mathbf{M}'))(\forall v \in O(\mathbf{M}'))((u, v) \neq (s, o) \rightarrow M'[u, v] = M[u, v])$;

2) *delete r from (s, o)* – удаление права доступа r из ячейки матрицы (s, o) ; в результате выполнения этого оператора матрица \mathbf{M} преобразуется в матрицу \mathbf{M}' , строки и столбцы которой помечены так же, как строки и столбцы матрицы \mathbf{M} , т.е. $\mathbf{S}(\mathbf{M}') = \mathbf{S}(\mathbf{M})$ и $\mathbf{O}(\mathbf{M}') = \mathbf{O}(\mathbf{M})$, $\mathbf{M}'[s, o] = \mathbf{M}[s, o] \setminus \{r\}$ и $(\forall u \in S(\mathbf{M}'))(\forall v \in O(\mathbf{M}'))((u, v) \neq (s, o) \rightarrow M'[u, v] = M[u, v])$;

3) *create subject s* – создание нового субъекта s ; в результате выполнения этого оператора матрица \mathbf{M} преобразуется в матрицу \mathbf{M}' , устроенную следующим образом: $\mathbf{S}(\mathbf{M}') = \mathbf{S}(\mathbf{M}) \cup \{s\}$ и $\mathbf{O}(\mathbf{M}') = \mathbf{O}(\mathbf{M}) \cup \{s\}$, $(\forall v \in O(\mathbf{M}'))(M'[s, v] = \emptyset)$; $(\forall u \in S(\mathbf{M}'))(\forall v \in O(\mathbf{M}'))(u \neq s \rightarrow M'[u, v] = M[u, v])$;

4) *create object o* – создание нового объекта o ; в результате выполнения этого оператора матрица \mathbf{M} преобразуется в матрицу \mathbf{M}' , устроенную следующим образом: $\mathbf{S}(\mathbf{M}') = \mathbf{S}(\mathbf{M})$ и $\mathbf{O}(\mathbf{M}') = \mathbf{O}(\mathbf{M}) \cup \{o\}$, $(\forall u \in S(\mathbf{M}'))(M'[u, o] = \emptyset)$, $(\forall u \in S(\mathbf{M}'))(\forall v \in O(\mathbf{M}'))(v \neq o \rightarrow M'[u, v] = M[u, v])$;

5) *destroy subject s* – удаление субъекта s ; в результате выполнения этого оператора матрица \mathbf{M} преобразуется в матрицу \mathbf{M}' , устроенную следующим образом: $\mathbf{S}(\mathbf{M}') = \mathbf{S}(\mathbf{M}) \setminus \{s\}$ и $\mathbf{O}(\mathbf{M}') = \mathbf{O}(\mathbf{M}) \setminus \{s\}$, $(\forall u \in S(\mathbf{M}'))(\forall v \in O(\mathbf{M}'))(M'[u, v] = M[u, v])$;

6) *destroy object o* – удаление объекта o ; в результате выполнения этого оператора матрица \mathbf{M} преобразуется в матрицу \mathbf{M}' , устроенную следующим образом: $\mathbf{S}(\mathbf{M}') = \mathbf{S}(\mathbf{M})$ и $\mathbf{O}(\mathbf{M}') = \mathbf{O}(\mathbf{M}) \setminus \{o\}$, $(\forall u \in S(\mathbf{M}'))(\forall v \in O(\mathbf{M}'))(M'[u, v] = M[u, v])$.

Из примитивных операторов формируются **команды**, которые имеют структуру:

- 1) условие выполнения команды,
 - 2) набор примитивных операторов,
- т.е. команды имеют вид

command $C(x_1, \dots, x_n)$
if
 $r_1 \in M[x_{i_1}, x_{j_1}] \&$
 $r_2 \in M[x_{i_2}, x_{j_2}] \&$
 \dots
 $r_m \in M[x_{i_m}, x_{j_m}]$
then
 $op_1,$
 $op_2,$
 \dots
 op_k
end

где r_1, \dots, r_m — права доступа, а op_1, \dots, op_k — примитивные операторы. Условия в теле команды могут отсутствовать. В качестве пояснения рассмотрим несколько примеров.

- 1) *Команда создания файла.* Пользователь s создает файл f , получает право владения и, например, права чтения и записи

command *create fail* (s, f)
create object f ;
enter own into $M[s, f]$;
enter r into $M[s, f]$;
enter w into $M[s, f]$;
end

- 2) *Команда создания процесса.* Процесс p создает процесс q , получает права владения, чтения и записи и передает процессу q право на запись и чтение по отношению к самому себе

command *exec process* (p, q)
create subject q ;
enter own into $M[p, q]$;
enter r into $M[p, q]$;
enter w into $M[p, q]$;
enter r into $M[q, p]$;
enter w into $M[q, p]$;
end

- 3) *Команда передачи права чтения файла.* Владелец p файла f передает субъекту q право чтения этого файла

command *grant read* (p, q, f)
if $own \in M[p, f]$
then
enter r into $M[q, f]$;
end

4) *Команда передачи права записи в файл.* Владелец p файла f передает субъекту q право записи в этот файл

command *grant read* (p, q, f)
 if $own \in M[p, f]$
 then
 enter w **into** $M[q, f]$;
end

Система управления доступом определяется программой управления доступом P , состоящей из конечного набора **команд**.

Состоянием системы называется набор $Q = \langle S(M), O(M), M \rangle$.

После выполнения примитивного оператора op система из состояния $Q = \langle S(M), O(M), M \rangle$ переходит в состояние $Q' = \langle S(M'), O(M'), M' \rangle$. Как принято в математической логике этот переход выражается формулой вида $Q \vdash_{op} Q'$.

В результате выполнения команды

command $C(x_1, \dots, x_n)$
 if
 $r_1 \in M[x_{i_1}, x_{j_1}] \&$
 $r_2 \in M[x_{i_2}, x_{j_2}] \&$
 \dots
 $r_m \in M[x_{i_m}, x_{j_m}]$
 then
 $op_1,$
 $op_2,$
 \dots
 op_k
end

система из состояния $Q = \langle S(M), O(M), M \rangle$ переходит в состояние $Q^* = \langle S(M^*), O(M^*), M^* \rangle$. Этот переход выражается формулой вида $Q \vdash_{C(x_1, \dots, x_n)} Q^*$. При этом считается, что

1) либо хотя бы одно из условий команды $C(x_1, \dots, x_n)$ не выполнено и тогда $Q^* = Q$,

2) либо выполнены все условия команды и существует такая последовательность состояний $Q_0 = Q, Q_1, Q_2, \dots, Q_n = Q^*$, что при любом $1 \leq i \leq n$ $Q_{i-1} \vdash_{op_i} Q_i$.

Если система из состояния Q может быть переведена в состояние Q^* с помощью конечного числа выполнений команд программы P , то говорят, что программа P может перевести систему из состояния Q в состояние Q^* , и обозначают это формулой вида $Q \vdash_P Q^*$.

Состояние Q *небезопасно по отношению к данному праву доступа r* и программе P управления доступом, если существует такое состояние $Q^* = \langle S(M^*), O(M^*), M^* \rangle$, что $Q \vdash_P Q^*$, причем в некоторой ячейке $M^*[s^*, o^*]$ матрицы M^* содержится право r и либо это новая по отношению к матрице M

ячейка, т.е. либо $s^* \notin S(M)$, либо $o^* \notin O(M)$, либо $r \notin M[s^*, o^*]$, т.е. соответствующая ячейка матрицы M не содержала право r .

Теорема 5.1. *Невозможно создать алгоритм, который по произвольному состоянию $Q = \langle S(M), O(M), M \rangle$ системы, программе P управления доступом и праву r определяет, безопасна ли система по отношению к данному праву доступа r и программе P .*

До к а з а т е л ь с т в о. Для доказательства теоремы воспользуемся установленной выше алгоритмической неразрешимостью проблемы остановки (останова) для машин Тьюринга.

По произвольной машине Тьюринга T с внешним алфавитом $A_T = \{a_0, a_1, \dots, a_n\}$, внутренним алфавитом $Q_T = \{q_0, q_1, \dots, q_m\}$ и программой P_T и по произвольной ее начальной конфигурации

$$a_{t_1} \dots a_{t_{i-1}} q_j a_{t_i} \dots a_{t_p}$$

следующим образом построим систему AC с дискреционной политикой управления доступом:

множество субъектов S системы AC совпадает с множеством ее объектов O и находится во взаимно однозначном соответствии с множеством целых чисел (каждый субъект сопоставляется с ячейкой ленты машины Тьюринга T , которую мы считаем потенциально бесконечной в обе стороны), т.е.

$$S = \{\dots, s_{-2}, s_{-1}, s_0, s_1, s_2, \dots\},$$

множество прав доступа R задается равенством

$$R = \{a_0, a_1, \dots, a_n, q_0, q_1, \dots, q_m, own, begin, end\},$$

подразумеваемый смысл указанных прав будет ясен из дальнейшего.

По начальной конфигурации

$$K = a_{t_1} \dots a_{t_{i-1}} q_j a_{t_i} \dots a_{t_p}$$

построим матрицу $M_{T,K}$ управления доступом:

строки и столбцы матрицы $M_{T,K}$ помечаем субъектами $s_1, \dots, s_{i-1}, s_i, \dots, s_p$, (они соответствуют уже заполненным (введенным в рассмотрение) ячейкам ленты машины Тьюринга T ,

слово $a_{t_1} \dots a_{t_{i-1}} a_{t_i} \dots a_{t_p}$, записанное на ленте машины Тьюринга T , размещаем по главной диагонали матрицы $M_{T,K}$, т.е. в диагональные ячейки $M[s_1, s_1]$, $M[s_2, s_2]$, \dots , $M[s_{i-1}, s_{i-1}]$, $M[s_i, s_i]$, \dots , $M[s_p, s_p]$ матрицы $M_{T,K}$ помещаем права $a_{t_1}, a_{t_2}, \dots, a_{t_{i-1}}, a_{t_i}, \dots, a_{t_p}$,

чтобы отразить линейный характер расположения ячеек ленты машины Тьюринга T во все ячейки верхней побочной диагонали, т.е. в ячейки $M[s_1, s_2]$, $M[s_2, s_3]$, \dots , $M[s_{i-1}, s_i]$, $M[s_i, s_{i+1}]$, \dots , $M[s_{p-1}, s_p]$, помещаем право владения (наследования) *own*,

чтобы показать, что i -ая ячейка ленты машины обозревается считывающей головкой, находящейся в состоянии q_j , помещаем в ячейку $M[s_i, s_i]$ право q_j ,

чтобы показать, что субъект s_1 соответствует самой левой ячейке ленты машины Тьюринга T , помещаем в соответствующую ячейку $M[s_1, s_1]$ право *begin*,

чтобы показать, что субъект s_p соответствует самой правой ячейке ленты машины Тьюринга T помещаем в соответствующую ячейку $M[s_p, s_p]$ право *end*.

По программе P_T машины Тьюринга T строим программу P управления доступом системы AC :

команде вида $q_j a_i \rightarrow q_k a_l S$ из программы P_T машины Тьюринга T сопоставим следующую команду программы P управления доступом

```
command CS( $s$ )
  if
     $q_j \in M[s, s] \ \&$ 
     $a_i \in M[s, s]$ 
  then
    delete  $q_j$  from  $(s, s)$ ;
    delete  $a_i$  from  $(s, s)$ ;
    enter  $q_k$  into  $(s, s)$ ;
    enter  $a_l$  into  $(s, s)$ ;
end
```

команде вида $q_j a_i \rightarrow q_k a_l R$ из программы P_T машины Тьюринга T сопоставим две команды программы P управления доступом

```
command CR1( $s, s'$ )
  if
     $own \in M[s, s'] \ \&$ 
     $q_j \in M[s, s] \ \&$ 
     $a_i \in M[s, s]$ 
  then
    delete  $q_j$  from  $(s, s)$ ;
    delete  $a_i$  from  $(s, s)$ ;
    enter  $a_l$  into  $(s, s)$ ;
    enter  $q_k$  into  $(s', s')$ ;
end
```

```
command CR2( $s$ )
  if
     $end \in M[s, s] \ \&$ 
     $q_j \in M[s, s] \ \&$ 
     $a_i \in M[s, s]$ 
  then
    delete  $q_j$  from  $(s, s)$ ;
    delete  $a_i$  from  $(s, s)$ ;
    delete end from  $(s, s)$ ;
    enter  $a_l$  into  $(s, s)$ ;
    create subject  $s'$  ;
```

*enter own into (s, s');
enter end into (s', s');
enter a₀ into (s', s');
enter q_k into (s', s');*

end

команде вида $q_j a_i \rightarrow q_k a_l L$ из программы P_T машины Тьюринга T сопоставим две команды программы P управления доступом

command CL1(s', s)

if

own $\in M[s', s]$ &
 $q_j \in M[s, s]$ &
 $a_i \in M[s, s]$

then

*delete q_j from (s, s);
delete a_i from (s, s);
enter a_l into (s, s);
enter q_k into (s', s');*

end

command CL2(s)

if

begin $\in M[s, s]$ &
 $q_j \in M[s, s]$ &
 $a_i \in M[s, s]$

then

*delete q_j from (s, s);
delete a_i from (s, s);
delete **begin** from (s, s);
enter a_l into (s, s);
create subject s' ;
enter **own** into (s', s);
enter **begin** into (s', s');
enter a₀ into (s', s');
enter q_k into (s', s');*

end

Нетрудно понять, что машина Тьюринга T , начав работать в конфигурации

$$a_{t_1} \dots a_{t_{i-1}} q_j a_{t_i} \dots a_{t_p},$$

остановится, т.е. придет в состояние q_0 , тогда и только тогда, когда для построенной системы АС и программы P управления доступом в ней ее начальное состояние $Q_K = \langle S(M_{T,K}), O(M_{T,K}), M_{T,K} \rangle$ небезопасно по отношению к праву доступа q_0 и программе P . \square

Ограниченный вариант рассмотренной проблемы включен в список \mathcal{NP} -трудных проблем монографии М. Гэри и Д. Джонсона [15], там же отмечается, что вопрос о принадлежности этой задачи классу \mathcal{NP} открыт.

Надежность систем защиты.

УСЛОВИЕ. Заданы множество R "полномочий" или "прав доступа", множество O "объектов", множество $S \subseteq O$ "субъектов", для каждой упорядоченной пары (s, o) из $S \times O$ задано множество $P(s, o) \subseteq R$ "полномочий" или "прав доступа" субъекта s к объекту o , конечное множество команд, каждая из которых имеет вид:

$$\text{если } r_1 \in P(s_1, o_1) \& r_2 \in P(s_2, o_2) \& \dots \& r_n \in P(s_n, o_n), \text{ то} \\ C_1, C_2, \dots, C_m,$$

где каждое C_i – примитивный оператор вида

"ввести право r в $P(s, o)$ ",

"удалить r из $P(s, o)$ ".

ВОПРОС. Для произвольного выделенного права $r \in R$ требуется определить, существует ли такая последовательность команд из C , в результате выполнения которой в некоторое $P(s, o)$, не содержащее выделенного права r , это право вводится (происходит "утечка права r ")?

Если в число примитивных операторов добавить операторы вида

"создать субъект s ",

"создать объект o ",

"удалить субъект s ",

"удалить объект o ", то, как показано выше, задача становится алгоритмически неразрешимой.

В указанной монографии отмечается, что для монооперационных систем, т.е. для систем, команды которых содержат лишь по одному примитивному оператору, рассматриваемая задача \mathcal{NP} -полна и разрешима за полиномиальное время для каждой фиксированной системы.

§6. Несколько \mathcal{NP} -полных проблем

В настоящее время список \mathcal{NP} -полных и \mathcal{NP} -трудных проблем включает тысячи разнообразных задач из различных разделов математики и ее приложений. Приведем несколько близких по формулировке к рассмотренным в пособии проблемам \mathcal{NP} -полных проблем из монографии М. Гэри и Д. Джонсона "Вычислительные машины и трудно решаемые задачи", относящихся к теории языков и автоматов, теории графов, алгебре и теории чисел.

Ранее была установлена алгоритмическая неразрешимость известной проблемы – **Проблема соответствия Э. Поста**. Ее ограниченный вариант является \mathcal{NP} -полной проблемой.

Ограниченное соответствие Э. Поста

УСЛОВИЕ. Заданы конечный алфавит Σ , две последовательности A_1, A_2, \dots, A_m и B_1, B_2, \dots, B_m и натуральное число n .

ВОПРОС. Существует ли $k \leq n$ и такая последовательность индексов i_1, \dots, i_k , что

$$A_{i_1} \dots A_{i_k} = B_{i_1} \dots B_{i_k}?$$

При снятии ограничения $k \leq n$ получаем уже рассматривавшуюся ранее хорошо известную проблему – **Проблема соответствия Э. Поста**, которая, как было показано выше, является алгоритмически неразрешимой.

Приведем несколько примеров \mathcal{NP} -полных проблем, относящихся к **регулярным и автоматным языкам** и связанных с рассматривавшимися выше вопросами.

Подстановка регулярного выражения

УСЛОВИЕ. Заданы два конечных алфавита $\Sigma = \{a_1, \dots, a_n\}$ и $X = \{x_1, \dots, x_m\}$, регулярное выражение E в алфавите $\Sigma \cup X$, регулярные выражения E_1, \dots, E_n в алфавите Σ и слово g в этом же алфавите.

ВОПРОС. Существуют ли в языках $L(E), L(E_1), \dots, L(E_n)$ такие слова w, v_1, \dots, v_n , что если в слове w заменить x_1 на v_1, \dots, x_n на v_n , то получим слово g ?

Проблема остается \mathcal{NP} -полной, если в качестве выражения E взять выражение, определяющее язык $L(E)$, состоящий из одного слова w , а в качестве выражений E_1, \dots, E_n взять такие выражения, что определяемые ими языки $L(E_1), \dots, L(E_n)$ являются универсальным языком Σ^* , т.е. на слова v_1, \dots, v_n не накладывается никаких ограничений.

Вопрос о принадлежности следующей проблемы классу \mathcal{NP} в настоящее время остается открытым.

Неэквивалентность регулярных выражений

УСЛОВИЕ. Заданы два регулярных выражения E_1 и E_2 в конечном алфавите Σ с операторами \cup, \cdot и $*$.

ВОПРОС. Верно ли, что языки $L(E_1)$ и $L(E_2)$, определяемые этими выражениями, различны?

Эта задача является $\mathcal{P} - \mathcal{SPACE}$ -полной.

Неуниверсальность регулярного выражения

По регулярному выражению E требуется определить, задает ли оно язык, отличный от универсального Σ^* .

Эта задача является $\mathcal{P} - \mathcal{SPACE}$ -полной.

Не содержащие звездочек регулярные выражения над алфавитом Σ и задаваемые (описываемые) ими языки определяются аналогично тому, как были определены регулярные выражения и задаваемые ими языки. Язык, определяемый регулярным выражением α , будем обозначать через $L(\alpha)$. При этом, если регулярное выражение α имеет вид (β) , то вместо $L((\beta))$ будем писать просто $L(\beta)$.

1) \emptyset, ε и a для любого символа a алфавита Σ являются не содержащими звездочек регулярными выражениями над алфавитом Σ и задают соответственно языки $L(\emptyset) = \emptyset, L(\varepsilon) = \{\varepsilon\}$ и $L(a) = \{a\}$.

2) Если α и β – не содержащие звездочек регулярные выражения над алфавитом Σ , то $(\alpha \cdot \beta)$ и $(\alpha \cup \beta)$ – не содержащие звездочек регулярные выражения над алфавитом Σ и $L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$ и $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$.

Неэквивалентность не содержащих звездочек регулярных выражений

По двум не содержащим звездочек регулярным выражениям E_1 и E_2 требуется определить, задают ли они различные языки.

Минимальное разделяющее регулярное выражение

УСЛОВИЕ. Заданы два конечных множества слов U и V в конечном алфавите Σ и натуральное число n .

ВОПРОС. Существует ли регулярное выражение длины не более n над алфавитом Σ такое, что $U \subseteq L(E)$ и $V \subseteq \Sigma^* \setminus L(E)$?

Минимальный разделяющий конечный автомат

УСЛОВИЕ. Заданы два конечных множества слов U и V в конечном алфавите Σ и натуральное число n .

ВОПРОС. Существует ли такой детерминированный конечный автомат M с n состояниями и с входным алфавитом Σ , что $U \subseteq L(M)$ и $V \subseteq \Sigma^* \setminus L(M)$?

Пересечение для конечных автоматов

УСЛОВИЕ. Заданы n конечных детерминированных автоматов M_1, \dots, M_n с одним и тем же входным алфавитом Σ .

ВОПРОС. Существует ли слово, принадлежащее всем языкам $L(M_1), \dots, L(M_n)$, т.е. будет ли непустым пересечение языков $\bigcap_{i=1}^n L(M_i)$?

При каждом фиксированном n задача полиномиально разрешима.

Выполнимость

УСЛОВИЕ. Заданы множество переменных X и конечный набор K дизъюнкций от этих переменных и их отрицаний.

ВОПРОС. Существует ли набор истинностных значений переменных из X , на котором все дизъюнкции из K истинны?

3-Выполнимость

УСЛОВИЕ. Заданы множество переменных X и конечный набор K таких дизъюнкций от этих переменных и их отрицаний, что в каждую из них входит в точности 3 литерала (литерал – это переменная или ее отрицание).

ВОПРОС. Существует ли набор истинностных значений переменных из X , на котором все дизъюнкции из K истинны?

3-Выполнимость при различных литералах

УСЛОВИЕ. Заданы множество переменных X и конечный набор K таких дизъюнкций от этих переменных и их отрицаний, что в каждую из них входит в точности 3 литерала.

ВОПРОС. Существует ли набор истинностных значений переменных из X , на котором все дизъюнкции из K истинны и при этом в каждой дизъюнкции найдется хотя бы один ложный литерал?

3-Выполнимость при одном истинном литерале

УСЛОВИЕ. Заданы множество переменных X и конечный набор K таких дизъюнкций от этих переменных и их отрицаний, что в каждую из них входит в точности 3 литерала.

ВОПРОС. Существует ли набор истинностных значений переменных из X , на котором все дизъюнкции из K истинны и при этом в каждой дизъюнкции истинен лишь один литерал?

Максимальная 2-выполнимость

УСЛОВИЕ. Заданы множество переменных X и конечный набор K таких дизъюнкций от этих переменных и их отрицаний, что в каждую из них входит в точности 2 литерала и число $n \leq |K|$.

ВОПРОС. Существует ли набор истинностных значений переменных из X , на котором истинны по крайней мере n дизъюнкций из K ?

При $n = |K|$ задача полиномиально разрешима.

Минимальная дизъюнктивная нормальная форма

УСЛОВИЕ. Заданы множество переменных X , конечный набор K истинностных значений этих переменных и натуральное число n .

ВОПРОС. Существует ли дизъюнктивная нормальная форма от этих переменных, содержащая не более n дизъюнктивных членов и истинная лишь на наборах из множества K ?

Вопрос о принадлежности следующей проблемы классу \mathcal{NP} в настоящее время остается открытым.

Теория первого порядка с равенством

УСЛОВИЕ. Задана замкнутая формула F теории первого порядка с равенством.

ВОПРОС. Верно ли, что формула F истинна во всех моделях теории первого порядка с равенством?

Доказательство принадлежности классу \mathcal{NP} следующей задачи достаточно сложно.

Позитивная логика предикатов

УСЛОВИЕ. Задана позитивная замкнутая формула F логики предикатов с равенством.

ВОПРОС. Верно ли, что формула F тождественно истинна?

Минимальное множество аксиом

УСЛОВИЕ. Заданы некоторый конечный алфавит Σ , конечное множество U слов в этом алфавите, конечноместное правило вывода Q на множестве U (n -местное правило вывода Q на множестве U – это просто n -местное отношение на множестве U , т.е. $R \subseteq U^n$), конечное множество $S \subseteq U$ слов в алфавите Σ и натуральное число m .

ВОПРОС. Существует ли подмножество S_0 множества S такое, что $|S_0| \leq m$ и все слова множества S выводимы из S_0 с помощью правила вывода Q ?

Сжатие и представление информации**Кратчайшая общая надпоследовательность**

УСЛОВИЕ. Заданы конечный алфавит Σ , конечное множество S слов в алфавите Σ и натуральное число n .

ВОПРОС. Существует ли слово w длины не более, чем n в алфавите Σ такое, что любое слово v из множества слов S является подпоследовательностью слова w , т.е. если v – это слово $a_1 \dots a_t$, то найдутся такие слова u_0, u_1, \dots, u_t в алфавите Σ , что $w = u_0 a_1 u_1 a_2 u_2 \dots a_t u_t$?

Кратчайшее общее "надслово"

УСЛОВИЕ. Заданы конечный алфавит Σ , конечное множество S слов в алфавите Σ и натуральное число n .

ВОПРОС. Существует ли слово w длины не более, чем n в алфавите Σ такое, что любое слово v из множества слов S является подсловом слова w , т.е. найдутся такие слова w_0 и w_1 в алфавите Σ , что $w = w_0 v w_1$?

Общая подпоследовательность наибольшей длины

УСЛОВИЕ. Заданы конечный алфавит Σ , конечное множество S слов в алфавите Σ и натуральное число n .

ВОПРОС. Существует ли слово w длины не менее чем n в алфавите Σ такое, что слово w является подсловом любого слова v из множества слов S ?

Представляющее слово

УСЛОВИЕ. Задано конечное множество S слов длины n в алфавите $\{0, 1\}$.

ВОПРОС. Существует ли слово w длины n в алфавите $\{0, 1\}$ такое, что для любого слова v из множества слов S найдется такое i ($1 \leq i \leq n$), что i -ые символы слов w и v совпадают?

Редактирование слова

УСЛОВИЕ. Заданы конечный алфавит Σ , два слова w и v в этом алфавите и натуральное число n .

ВОПРОС. Можно ли из слова w получить слово v с помощью не более n операций вычеркивания символа и перестановки соседних символов?

Приведем несколько примеров \mathcal{NP} -полных и \mathcal{NP} -трудных проблем, относящихся к разделу **Алгебра и теория чисел**.

Квадратичные сравнения

УСЛОВИЕ. Заданы натуральные числа a , b и c .

ВОПРОС. Существует ли натуральное число d такое, что

$$d^2 \equiv a \pmod{b} \text{ и } d < c?$$

Сложность этой задачи используется в некоторых криптопротоколах.

Следующая проблема тесно связана с хорошо известной *Китайской теоремой об остатках*.

Система сравнений

УСЛОВИЕ. Задан набор $(a_1, m_1), \dots, (a_n, m_n)$ пар натуральных чисел.

ВОПРОС. Существует ли целое число d такое, что

$$\bigwedge_{i=1}^n d \equiv a_i \pmod{m_i}?$$

Вопрос о принадлежности классу \mathcal{NP} следующей задачи открыт. Известна ее \mathcal{NP} -трудность и принадлежность классу \mathcal{NP} при любом фиксированном n .

При любом фиксированном $n \geq 5$ задача \mathcal{NP} -полна.

Обобщение этой задачи на кольца вида $Z[\sqrt{d}]$ при некоторых d дает алгоритмически неразрешимую проблему.

Совместная делимость линейных полиномов

УСЛОВИЕ. Задан набор $(a_1, b_1), \dots, (a_n, b_n)$ пар векторов

$$a_i = (a_i[0], a_i[1], \dots, a_i[m])$$

и

$$b_i = (b_i[0], b_i[1], \dots, b_i[m])$$

$(1 \leq i \leq n)$ с натуральными компонентами.

ВОПРОС. Существуют ли такие натуральные числа x_1, \dots, x_m , что

$$\bigwedge_{i=1}^n (a_i[0] + a_i[1]x_1 + \dots + a_i[m]x_m) \mid (b_i[0] + b_i[1]x_1 + \dots + b_i[m]x_m)?$$

Эта задача – частный случай вопроса о разрешимости универсальной теории множества натуральных чисел в сигнатуре, содержащей функциональный символ $+$ для сложения и предикатный символ \mid для отношения делимости. Разрешимость этой проблемы установлена А.П. Бельтюковым в статье [5].

Сравнительная делимость

УСЛОВИЕ. Задан набор $(a_1, b_1), \dots, (a_n, b_n)$ пар натуральных чисел.

ВОПРОС. Существует ли такое натуральное число d , что число чисел a_i , делящихся на d , больше числа чисел b_j , делящихся на d ?

Делимость показательного выражения

УСЛОВИЕ. Задан набор $(a_1, b_1), \dots, (a_n, b_n)$ пар натуральных чисел и целое число d .

ВОПРОС. Делится ли число $\prod_{i=1}^n (d^{b_i} - 1)$ на число $\prod_{i=1}^n (d^{a_i} - 1)$?

Делимость многочленов

УСЛОВИЕ. Задан набор $(a_1, b_1), \dots, (a_n, b_n)$ пар натуральных чисел.

ВОПРОС. Делится ли многочлен $\prod_{i=1}^n (x^{b_i} - 1)$ на многочлен $\prod_{i=1}^n (x^{a_i} - 1)$?

Доказательство принадлежности следующей задачи классу \mathcal{NP} нетривиально.

Неделимость произведения многочленов

УСЛОВИЕ. Заданы последовательности

$$\langle (a_i[1], b_i[1]), \dots, a_i[m], b_i[m] \rangle$$

$(1 \leq i \leq n)$ пар целых чисел, причем при любых i и j : $b_i[j] \geq 0$, и натуральное число N .

ВОПРОС. Верно ли, что многочлен $\prod_{i=1}^n \left(\sum_{j=1}^m a_i[j] x^{b_i} \right)$ не делится на многочлен $x^N - 1$?

Вопрос о принадлежности следующей задачи классу \mathcal{NP} открыт.

Нетривиальный наибольший общий делитель многочленов

УСЛОВИЕ. Заданы последовательности

$$\langle (a_i[1], b_i[1]), \dots, a_i[m], b_i[m] \rangle$$

($1 \leq i \leq n$) пар целых чисел, причем при любых i и j : $b_i[j] \geq 0$.

ВОПРОС. Верно ли, что наибольший общий делитель многочленов $\sum_{j=1}^m a_i[j] x^{b_i[j]}$ имеет положительную степень, т.е. эти многочлены не являются взаимно простыми?

Рассмотрим несколько \mathcal{NP} -полных проблем, связанных с **разрешимостью уравнений**.

Квадратные диофантовы уравнения

УСЛОВИЕ. Заданы натуральные числа a , b и c .

ВОПРОС. Существуют ли натуральные числа d и n такие, что $ad^2 + bn = c$?

Проблема разрешимости в целых или натуральных числах уравнений вида

$$F(x_1, \dots, x_n) = 0,$$

где $F(x_1, \dots, x_n)$ – произвольный полином с целыми коэффициентами, алгоритмически неразрешима [48].

Алгебраические уравнения в поле Z_2

УСЛОВИЕ. Заданы полиномы $P_i(x_1, \dots, x_n)$ ($1 \leq i \leq m$) над полем Z_2 .

ВОПРОС. Разрешима ли в поле Z_2 система уравнений

$$\bigwedge_{i=1}^m P_i(x_1, \dots, x_n) = 0?$$

Можно считать, что все полиномы имеют степень не более двух.

При $m = 1$ задача полиномиально разрешима.

Метод Гаусса дает полиномиальный алгоритм для систем линейных уравнений.

Корни на единичной окружности многочленов

УСЛОВИЕ. Задана последовательность

$$\langle (a[1], b[1]), \dots, a[m], b[m] \rangle$$

пар целых чисел, причем при любом j : $b[j] \geq 0$, и натуральное число N .

ВОПРОС. Верно ли, что многочлен $\sum_{j=1}^m a[j] x^{b[j]}$ имеет корень на единичной окружности?

Вопрос о принадлежности следующих трех задач классу \mathcal{NP} открыт.

Число корней произведения многочленов

УСЛОВИЕ. Заданы последовательности

$$\langle (a_i[1], b_i[1]), \dots, a_i[m], b_i[m] \rangle$$

$(1 \leq i \leq n)$ пар целых чисел, причем при любых i и j : $b_i[j] \geq 0$, и натуральное число N .

ВОПРОС. Верно ли, что многочлен $\prod_{i=1}^n \left(\sum_{j=1}^m a_i[j] x^{b_i} \right)$ имеет менее N различных комплексных корней?

Периодическое решение рекуррентного уравнения

УСЛОВИЕ. Задана последовательность

$$\langle (a[1], b[1]), \dots, (a[m], b[m]) \rangle$$

$(1 \leq i \leq n)$ пар целых чисел, причем при любом j : $b[j] > 0$.

ВОПРОС. Существует ли такая последовательность целых чисел c_0, c_1, \dots, c_{n-1} ($n \geq \max\{b_1, \dots, b_m\}$), что бесконечная последовательность целых чисел $c_0, c_1, \dots, c_{n-1}, \dots$, определяемая рекуррентным уравнением

$$c_i = \sum_{j=1}^m a[j] \cdot c_{(i-b[j])}$$

при всех $i \geq n$, удовлетворяет равенству $c_i = c_{i \pmod n}$, т.е. является периодической?

Вычисление перманента

УСЛОВИЕ. Заданы матрица M порядка n с элементами $0, 1$ и натуральное число $N \leq n!$.

ВОПРОС. Верно ли, что перманент матрицы M равен N ?

Унификация с коммутативными операторами

УСЛОВИЕ. Заданы множество переменных V , множество констант C и множество $(E_1, F_1), \dots, (E_n, F_n)$ упорядоченных пар термов. (Каждая переменная v из V и каждая константа c из C – терм, если t и s – термы, то $(t + s)$ – терм).

ВОПРОС. Можно ли каждой переменной v сопоставить замкнутый терм $I(v)$, т.е. терм без переменных, так, чтобы выполнялись тождества $I(E_1) \equiv I(F_1), \dots, I(E_n) \equiv I(F_n)$? (Для терма t через $I(t)$ обозначается терм, полученный из терма t заменой каждой входящей в него переменной v на замкнутый терм $I(v)$, а отношение тождества \equiv для замкнутых термов определяется индукцией по их построению: $t \equiv s$, если либо t есть s , либо $t = (\alpha + \beta)$, $s = (\gamma + \delta)$ и либо $\alpha \equiv \gamma$ и $\beta \equiv \delta$, либо $\alpha \equiv \delta$ и $\beta \equiv \gamma$.)

Приведенный достаточно краткий список \mathcal{NP} -полных проблем ни в коей мере не претендует на какую-либо полноту. При его формировании преследовалась лишь одна цель – показать сколь разнообразны \mathcal{NP} -полные проблемы и как они тесно связаны с вопросами, рассматривавшимися в различных математических дисциплинах.

ПОСЛЕСЛОВИЕ

В планах автора написание продолжения пособия, в котором предполагается более глубоко рассмотреть некоторые фундаментальные алгоритмические проблемы алгебры и теории чисел – фундаментальные теоремы С.П. Новикова о неразрешимости проблемы равенства для групп, С.И. Адяна о нераспознаваемости нетривиальных свойств групп по их заданиям, М. Дэвиса, Х. Путнам, Дж. Робинсон и Ю.В. Матиясевича о неразрешимости проблемы совместности для диофантовых уравнений и ряд приложений этих результатов для установления алгоритмической неразрешимости некоторых проблем топологии, математического анализа и теории обыкновенных дифференциальных уравнений.

Литература

- [1] Адян С.И. *Алгоритмическая неразрешимость проблем распознавания некоторых свойств групп* // ДАН СССР. - 1955. - Том 103, № 4. - С. 533 – 535.
- [2] Адян С.И. *Неразрешимость некоторых алгоритмических проблем теории групп* // Тр. Моск. матем. о-ва. - 1957. - Том 6. - С. 231 – 298.
- [3] Адян С.И., Дурнев В.Г. *Алгоритмические проблемы для групп и полугрупп* // Успехи матем. наук. - 2000. - Том 55, № 2. - С. 3 – 94.
- [4] Ахо А., Хопкрофт Дж., Ульман Дж. *Построение и анализ вычислительных алгоритмов*. М.: Мир, 1983.
- [5] Бельтюков А.П. *Разрешимость универсальной теории натуральных чисел со сложением и делимостью* // Записки научн. семинаров Ленингр. отд. Матем. ин-та. АН СССР. Л., 1976. - Том 60, № 7. - С. 15 – 28.
- [6] Березнюк С.Л. *Лекции по теории алгоритмов*. Новосибирск: НГУ, 2000.
- [7] Бокуть Л.А., Кукин Г.П. *Неразрешимые алгоритмические проблемы для полугрупп, групп и колец*. Итоги науки и техники. ВИНТИ. Алгебра. Топология. Геометрия. 1987. - Том 25. - С. 3 – 65.
- [8] Борисов В.В. *Простые примеры групп с неразрешимой проблемой тождества* // Матем. заметки. - 1969. - Том 6, вып. 5. - С. 521 – 532.
- [9] Булос Дж., Джеффри Р. *Вычислимость и логика*. М.: Мир, 1994.
- [10] Валиев М.К. *Примеры универсальных конечно определенных групп* // ДАН СССР. - 1973. - Том 211. - С. 265 – 268.
- [11] Володин И.А., Кузнецов В.Е., Фоменко А.Т. *О проблеме алгоритмического распознавания стандартной трехмерной сферы* // Успехи матем. наук. - 1974. - Том 29, № 5. - С. 71 – 168.
- [12] Гинзбург С. *Математическая теория контекстно-свободных языков*. М.: Мир, 1970.
- [13] Гладкий А.В. *Формальные грамматики и языки*. М.: Наука, 1973.

-
-
- [14] Гольберг А.И. *О невозможности усиления некоторых результатов Гриндлингера и Линдона* // Успехи матем. наук. - 1978. Том 33, № 6. - С. 201 – 202.
- [15] Гэри М., Джонсон Д. *Вычислительные машины и труднорешаемые задачи*. М.: Мир, 1979.
- [16] Дехтярь М.И. *Сложность решения одного класса уравнений в словах* // VII Всесоюзная конференция по мат. логике. Тезисы докладов. - Новосибирск. - 1984. - С. 58.
- [17] Дурнев В.Г. *Позитивная теория свободной полугруппы* // ДАН СССР. - 1973. - Том 211, № 4. - С. 772 – 774.
- [18] Дурнев В.Г. *О позитивных формулах на свободных полугруппах* // Сиб. матем. журн. - 1974. Том 25, № 5. - С. 1131 – 1137.
- [19] Дурнев В.Г. *Об уравнениях на свободных полугруппах и группах* // Матем. заметки. - 1974. - Том 16, № 5. - С. 717 – 724.
- [20] Дурнев В.Г. *О системах уравнений на свободных нильпотентных группах* // Вопросы теории групп и гомолог. алгебры. - Ярославль: ЯрГУ, 1981. - С. 66 – 69.
- [21] Дурнев В.Г. *Неразрешимость позитивной $\forall\exists^3$ -теории свободной полугруппы* // Сиб. матем. журн. - 1995. - Том 36, № 5. - С. 1067 – 1080.
- [22] Ершов Ю.Л. *Новые примеры неразрешимых теорий* // Алгебра и логика. - 1966. - Том 5, № 5. - С. 37 – 48.
- [23] Ершов Ю.Л. *Об элементарных теориях групп* // ДАН СССР. - 1972. - Том 203, № 6. - С. 1240 – 1243.
- [24] Катленд Н. *Вычислимость. Введение в теорию рекурсивных функций*. М.: Мир, 1983.
- [25] Клини С.К. *Введение в метаматематику*. М.: ИЛ, 1957.
- [26] Клини С.К. *Математическая логика*. М.: Мир, 1973.
- [27] Колмогоров А.Н., Успенский В.А. *К определению понятия алгоритма* // Успехи матем. наук. - 1958. - Том 13, вып. 4. - С. 3 – 28.
- [28] Косовский Н.К. *Некоторые свойства решений уравнений в свободной полугруппе* // Записки научн. семинаров Ленингр. отд. Матем. ин-та. АН СССР. - Л. - 1972. - Том 32. - С. 21 – 28.
- [29] Косовский Н.К. *О множествах, представимых в виде решений уравнений в словах и длинах* // Вторая всесоюзная конфер. по матем. логике. Тезисы кратких сообщений. - М. - 1972. - С. 23.

- [30] Косовский Н.К. *О решении систем, состоящих одновременно из уравнений в словах и неравенств в длинах слов* // Записки научн. семинаров Ленингр. отд. Матем. ин-та. АН СССР. - Л. - 1973. - Том 33. - С. 24 – 29.
- [31] Косовский Н.К. *Элементы математической логики и ее приложения к теории субрекурсивных предикатов*. - Л.: Изд-во. ЛГУ, 1981.
- [32] Лавров И.А., Максимова Л.Л. *Задачи по теории множеств, математической логике и теории алгоритмов*. - М.: Наука, 1975.
- [33] Маканин Г.С. *Проблема разрешимости уравнений в свободной полугруппе* // Матем. сб. - 1977. - Том 103(145), № 2(6). - С. 147 – 236.
- [34] Маканин Г.С. *Уравнения в свободной группе* // Изв. АН СССР. Сер. матем. - 1982. - Том 46, № 6. - С. 1199 – 1273.
- [35] Маканин Г.С. *Универсальная теория и позитивная теория свободной группы* // Изв. АН СССР. Сер. матем. - 1984. - Том 48, № 4. - С. 29 – 46.
- [36] Мальцев А.И. *О свободных разрешимых группах* // ДАН СССР. - 1960. - Том 130, № 3. - С. 495 – 498.
- [37] Мальцев А.И. *Алгоритмы и рекурсивные функции*. М.: Наука, 1986.
- [38] Манин Ю.И. *Вычислимое и невычислимое*. М.: Советское радио, 1979.
- [39] Марков А.А. *Невозможность некоторых алгоритмов в теории ассоциативных систем* // ДАН СССР. - 1947. - Том 55, № 7. - С. 587 – 590.
- [40] Марков А.А. *Теория алгоритмов*. Тр. МИАН. - 1951. - Том 38. - С. 176 – 189.
- [41] Марков А.А. *Теория алгоритмов*. М.: Наука. - 1954. Труды МИАН. - Том 42.
- [42] Марков А.А. *Неразрешимость проблемы гомеоморфии* // ДАН СССР. - 1958. - Том 121, № 2. - С. 218 – 220.
- [43] Марков А.А. *К проблеме представимости матриц* // Z. Math. Log. und Grundle. Math. - 1958. - Том 4, № 2. - С. 157 – 168.
- [44] Марков А.А., Нагорный Н.М. *Теория алгоритмов*. М.: Наука, 1984.
- [45] Марченков С.С. *Неразрешимость позитивной $\forall\exists$ -теории свободной полугруппы* // Сиб. матем. журн. - 1982. - Том 32, № 1. - С. 196 – 198.
- [46] Матвеев С.В. *Алгоритм распознавания трехмерной сферы (по А. Томпсон)* // Матем. сборник. - 1995. - Том 186, № 5. - С. 69 – 84.
- [47] Матиясевич Ю.В. *Простые примеры неразрешимых ассоциативных исчислений* // ДАН СССР. - 1967. - Том 173, № 6. - С. 1264 – 1266.

-
-
- [48] Матиясевич Ю.В. *Диофантовость перечислимых множеств* // ДАН СССР. - 1970. - Том 130, № 3. - С. 495 – 498.
- [49] Матиясевич Ю.В. *Десятая проблема Гильберта*. М.: Наука, 1993.
- [50] Мендельсон Э. *Введение в математическую логику*. М.: Наука, 1976.
- [51] Мерзляков Ю.И. *Позитивные формулы на свободных группах* // Алгебра и логика. - 1966. - Том 5, вып. 4. - С. 25 – 42.
- [52] Михайлова К.А. *Проблема вхождения для прямых произведений групп* // ДАН СССР. - 1958. - Том 119. - С. 1103 – 1105.
- [53] Михайлова К.А. *Проблема вхождения для свободных произведений групп* // ДАН СССР. - 1959. - Том 127. - С. 746 – 748.
- [54] Морозов А.С. *Лекции по теории алгоритмов*. Новосибирск: НГУ, 2000.
- [55] Новиков П.С. *Об алгоритмической неразрешимости проблемы тождества теории групп* // ДАН СССР. - 1952. - Том 85, № 4. - С. 709 – 712.
- [56] Новиков П.С. *Неразрешимость проблемы сопряженности в теории групп* // Изв. АН СССР. Сер. матем. - 1954. - Том 18, № 6. - С. 485 – 524.
- [57] Новиков П.С. *Об алгоритмической неразрешимости проблемы тождества слов в теории групп* // М.: Наука, 1955. Труды МИАН. - Том 44.
- [58] Новиков П.С., Адян С.И. *Проблема тождества для полугрупп с односторонним сокращением* // Ztschr. math. Log. und Grundle. Math. - 1958. - Bb. 4. - S. 66 – 88.
- [59] Новиков П.С., Адян С.И. *Определяющие соотношения и проблема тождества для свободных периодических групп нечетного порядка* // Изв. АН СССР. Сер. матем. - 1968. - Том 32, № 4. - С. 971 – 979.
- [60] Новиков П.С., Адян С.И. *О коммутативных подгруппах и проблеме сопряженности в свободных периодических группах нечетного порядка* // Изв. АН СССР. Сер. матем. - 1968. - Том 32, № 5. - С. 1176 – 1190.
- [61] Новиков П.С. *Элементы математической логики*. М.: Наука, 1973.
- [62] Павлов Р.Д. *К проблеме распознавания групповых свойств* // Матем. заметки. - 1971. - Том 10, № 2. - С. 169 – 180.
- [63] Павлов Р.Д. *О проблеме распознавания групповых свойств в ограниченных алфавитах* // Сердика Бълг. мат. списание. - 1979. - Том 5, № 3. - С. 252 – 271.
- [64] Пападимитриу Х., Стайглиц К. *Комбинаторная оптимизация. Алгоритмы и сложность*. М.: Мир, 1985.

- [65] Ремесленников В.Н., Романьков В.А. *Теоретико-модельные и алгоритмические вопросы теории групп*. Итоги науки и техники. ВИНТИ. Алгебра. Топология. Геометрия. - 1983. - Том 21. - С. 3 – 79.
- [66] Репин Н.Н. *Об уравнениях в нильпотентных группах* // IX Всесоюзный симпозиум по теории групп. Тезисы докладов. - Москва. - 1984. - С. 55.
- [67] Репин Н.Н. *Некоторые просто заданные группы, для которых невозможен алгоритм, распознающий разрешимость уравнений* // Вопросы кибернетики. - 1988. - Вып. 134. - С. 176 – 175.
- [68] Репин Н.Н. *Уравнения с одной неизвестной в нильпотентной группе* // Мат. заметки. - 1983. - Том 34, № 2. - С. 201 – 206.
- [69] Роджерс Х. *Теория рекурсивных функций и эффективная вычислимость*. М.: Мир, 1972.
- [70] Романовский Н.С. *О некоторых алгоритмических проблемах для разрешимых групп* // Алгебра и логика. - 1974. - Том 13, № 1. - С. 26 – 34.
- [71] Романьков В.А. *О неразрешимости проблемы эндоморфной сводимости в свободных нильпотентных группах и в свободных кольцах* // Алгебра и логика. - 1977. - Том 16, № 4. - С. 457 – 471.
- [72] Романьков В.А. *Об уравнениях в свободных метабелевых группах* // Сиб. матем. журн. - 1979. - Том 20, № 3. - С. 671 – 673.
- [73] Романьков В.А. *Об универсальной теории нильпотентных групп* // Матем. заметки. - 1979. - Том 25, № 4. - С. 487 – 496.
- [74] Саркисян Р.А. *Алгоритмические вопросы для линейных алгебраических групп I, II*. // Матем. сборник. - 1980. - Том 113, № 2. - С. 179 – 216; № 3. - С. 400 – 436.
- [75] Семенов А.Л. *Интерпретация свободных алгебр в свободных группах* // ДАН СССР. - 1980. - Том 252, № 6. - С. 1326 – 1332.
- [76] Слободской А.М. *Неразрешимость универсальной теории конечных групп* // Алгебра и логика. - 1981. - Том 20, № 2. - С. 207 – 230.
- [77] Тайманов А.Д., Хмелевский Ю.И. *Разрешимость универсальной теории свободной полугруппы* // Сиб. мат. журн. - 1980. - Том 21, № 1. - С. 228 – 230.
- [78] Тайцлин М.А. *О проблеме изоморфизма для коммутативных полугрупп* // Матем. сборник. - 1974. - Том 93, № 1. - С. 103 – 128.
- [79] Тайцлин М.А. *Об алгоритмической проблеме для коммутативных полугрупп* // ДАН СССР. Сер. мат.-физ. - 1968. - Том 178, № 4. - С. 786 – 789.

- [80] Тайцлин⁴ Тайцлин М.А. *Проблема изоморфизма для коммутативных полугрупп решается положительно* // Теория моделей и ее применения. Алма-Ата, 1980. - С. 75 – 81.
- [81] Тартаковский В.А. *О проблеме тождества для некоторых типов групп* // ДАН СССР. - 1947. - Том 58. - С. 1909 – 1910.
- [82] Тартаковский В.А. *Решение проблемы тождества для группы с k -сократимым базисом при $k = 6$* // Изв. АН СССР. Сер. матем. - 1949. - Том 13. - С. 483 – 494.
- [83] Трахтенброт Б.А. *Алгоритмы и вычислительные автоматы*. М.: Советское радио, 1974.
- [84] Харлампович О.Г. *Конечно определенная разрешимая группа с неразрешимой проблемой равенства* // Изв. АН СССР. Сер. матем. - 1981. - Том 45, № 4. - С. 852 – 873.
- [85] Хмелевский Ю.И. *Решение уравнений в словах с тремя неизвестными* // ДАН СССР. - 1967. - Том 177, № 5. - С. 1023 – 1025.
- [86] Хмелевский Ю.И. *Уравнения в свободной полугруппе*. М.: Наука. 1971. (Тр. МИАН. Т.107).
- [87] Хмелевский Ю.И. *Системы уравнений в свободной группе*. I, II // Изв. АН СССР. Сер. мат. - 1971. - Том 35, № 6. - С. 1237 – 1268; 1972. - Том 36, № 1. - С. 110 – 179.
- [88] Хопкрофт Дж., Мотвани Р., Ульман Дж. *Введение в теорию автоматов, языков и вычислений* М.: Издат. дом “Вильямс”, 2002.
- [89] Шенфилд Дж. *Математическая логика*. М.: Наука, 1975.
- [90] Цейтин Г.С. *Относительно проблемы распознавания свойств ассоциативных исчислений* // ДАН СССР. - 1956. - Том 107, № 2. - С. 209 – 212.
- [91] Цейтин Г.С. *Ассоциативное исчисление с неразрешимой проблемой эквивалентности* // Труды матем. ин-та. АН СССР. - 1958. - Том 52. - С. 172 – 189.
- [92] Эббинхауз Г.Д., Якобс К., Ман Ф.К., Хермес Г. *Машины Тьюринга и рекурсивные функции*. М.: Мир, 1972.
- [93] Aanderaa S. *A proof of Higman's embedding theorem using Britton extentions of groups* // Word Problems I. Studies in Logic and the Foundations of Mathematics. - 1973.
- [94] Boone W.W. *Certain simple unsolvable problems in group theory* // Proc. Kon. ned. akad. wetensch. - A, 1957. - Vol. 60. - P. 22 – 27, 227 – 232.

-
-
- [95] Boone W.W. *The word problem* // Ann. Math. - 1959. - Vol. 70, № 2. - P. 207 – 265.
 - [96] Boone W.W., Rogers H. *On a problem of J.H.C. Whitehead and a problem of Alonzo Church* // Math. Scand. - 1969. - Vol. 19, № 2. - P. 185 – 192.
 - [97] Boone W.W., Higman G. *An algebraic characterization of groups with soluble word problem* // J. Austral. Math. Soc. - 1974. - Vol. 18, № 1. - P. 41 – 53.
 - [98] Britton J.L. *The word problem for groups* // Proc. London Math. Soc. - 1958. - Vol. 8. - P. 493 – 506.
 - [99] Britton J.L. *The word problem* // Ann. Math. - 1963. - Vol. 77, № 1. - P. 16 – 32.
 - [100] Büchi J.R. and Senger S. *Definability in the existential theory of concatenation and undecidable extensions of this theory* // Z. Mat. Log. und Grundle. Math. - 1988. - Vol. 34, № 4. - P. 337 – 342.
 - [101] Büchi J.R. and Senger S. *Coding in the existential theory of concatenation* // Arch. math. Logik. 26 (1986 / 87). - P. 101 – 106.
 - [102] Church A. *An unsolvable problem of elementary number theory* // Amer. J. Math. - 1936. - Vol. 58, № 2. - P. 345 – 363.
 - [103] Church A. *A note on the Entscheidungsproblem* // J. Symbolic Logic. - 1936. - Vol. 1, № 1. - P. 40 – 41.
 - [104] Dehn M. *Über unendliche diskontinuierliche Gruppen* // Math. Ann. - 1911. - Bd. 71. - S. 116 – 144.
 - [105] Greendlinger M. *Dehn's algorithm for the word problem* // Comm. Pure and Appl. Math. - 1960. - Vol. 13. - P. 67 – 83.
 - [106] Greendlinger M. *On Dehn's algorithms for the conjugacy and word problems with applications* // Comm. Pure and Appl. Math. - 1960. - Vol. 13. - P. 641 – 677.
 - [107] Grunewald F., Segal D. *Some general algorithms. I. Arithmetic groups. II. Nilpotent groups* // Ann. Math. - 1980. - Vol. 112, № 3. - P. 531 – 583; P. 585 – 617.
 - [108] Hall M.Jr. *The word problem for semi-groups with two generators* // J. Symbolic Logic. - 1949. - Vol. 14. - P. 115 – 118.
 - [109] Hall P. *On the finiteness of certain soluble groups* // Proc. London Math. Soc. - 1959. - Vol. 9. - P. 592 – 622.
 - [110] Howie J., Pride S. *The word problem for one-relator semigroups* // Proc. of the Cambridge Phil. Soc. - 1986. - Vol. 99, № 1. - P. 33 – 44.

- [111] Higman G., Neumann B.H., Neumann H. *Embedding theorems for groups* // J. London Math. Soc. - 1949. - Vol. 24. - P. 247 – 254.
- [112] Higman G. *Subgroups of finitely presented groups* // Proc. Roy. Soc. London A. - 1961. - Vol. 262, № 1311. - P. 455 – 475.
- [113] Lipshitz L. *The Diophantine problem for addition and divisibility* // Trans. Amer. Math. Soc. - 1978. - Vol. 235. - P. 271 – 283.
- [114] Lipschutz S. *On Greendlinger groups* // Commun. Pure and Appl. Math. - 1970. - Vol. 23, № 5. - P. 743 – 747.
- [115] Lyndon R.C. *On Dehn's algorithm* // Math. Ann. - 1966. - Vol. 166. - P. 208 – 228.
- [116] Lyndon R.C. *Equations in free groups* // Trans. Amer. Math. Soc. - 1960. - Vol. 96. - P. 445 – 457.
- [117] Macintyre A. *On algebraically closed groups* // Ann. Math. - 1972. - Vol. 96, № 1. - P. 53 – 97.
- [118] Miller C.F. III. *Some connection between Hilbert's 10th problem and the theory of groups* // Word Probl. Decis. Probl. Group Theory. - Amsterdam; London. - P. 483 – 506.
- [119] Neumann B.H. *A note on algebraically closed groups* // J. London. Math. Soc. - 1952. - Vol. 27. - P. 227 – 242.
- [120] Neumann B.H. *The isomorphism problem for algebraically closed groups* // Studies in Logic and Foundation of Mathematics. - Vol. 95; Word Problems. North-Holland Publishing Company. - Amsterdam, 1973. - P. 553 – 562.
- [121] Nielsen J. *Om Regning med ikke-kommutative Faktoren og dens Anvendelse i Gruppeteorien* // Matematisk Tidskrift B. - 1921. - S. 77 – 94.
- [122] Post E. *Intoduction to a general theory of elementary propositions* // Amer. J. Math. - 1921. - Vol. 43.
- [123] Post E.L. *Finite combinatory processes – formulation 1* // Journal of Symbolic Logic. - 1936. - Vol. 1, № 3. - P. 103 – 105.
- [124] Post E.L. *A variant of a recursively unsolvable problem* // Bull. Amer. Math. Soc. - 1946. - Vol. 52. - P. 264 – 268.
- [125] Post E.L. *Recursive unsolvability of a problem of Thue* // J. Symbol Log. - 1947. - Vol. 12, № 1. - P. 1 – 11.
- [126] Quine W. *Concatenation as a basis for arithmetic* // J. Symbol Log. - 1946. - Vol. 11. - P. 105 – 114.

- [127] Rabin M.O. *Recursive unsolvability of group theoretic problems* // Ann. Math. - 1958. - Vol. 67, № 1. - P. 172 – 194.
- [128] Rips E. *Another characterization of finitely generated groups with a solvable word problem* // Bull. London. Math. Soc. - 1982. - Vol. 14, № 1. - P. 43 – 44.
- [129] Tarski A., Mostowski A., Robinson R.M. *Undecidable theories*. - NY, 1953. XI+ 98p.
- [130] Thue A. *Problem über Veränderungen von Zeichenreihen nach gegebenen Regeln* // Vid. Skr. Math.-natur. - KI, 1914. - № 10.
- [131] Tietze H. *Über topologischen Invarianten mehrdimensionaler Mannigfaltigkeiten* // Monatsh. Math. Phys. - 1908. - Vol. 19. - P. 1 – 118.
- [132] Thompson A. *Thin position and the recognition problem for S^3* // Preprint. - 1994.
- [133] Turing A.M. *On computable numbers, with an application to the Entscheidungsproblem* // Proceedings of London Mathematical Society. Ser. 2. - 1936. - Vol. 42, № 3, 4. - P. 230 – 265.
- [134] Turing A.M. *The word problem in semigroups with cancellation* // Ann. Math. - 1950. - Vol. 52. - P. 491 – 505.
- [135] Schupp P.E. *On the substitution problem for free groups* // Proc. Amer. Math. Soc. - 1969. - Vol. 23, № 2. - P. 421 – 423.
- [136] Scott D. *A short recursively unsolvable problem (abstract)* // J. Symbol. Log. - 1956. - Vol. 21, № 1. - P. 11 – 112.
- [137] Scott E.A. *A Finitely Presented Simple Group with Unsolvability Conjugacy Problem* // J. Algebra. - 1984. - Vol. 90, № 2. - P. 333 – 353.
- [138] Scott W.R. *Algebraically closed groups* // Proc. Amer. Math. Soc. - 1951. - Vol. 2. - P. 118 – 121.

УЧЕБНОЕ ИЗДАНИЕ

Дурнев Валерий Георгиевич

ЭЛЕМЕНТЫ ТЕОРИИ АЛГОРИТМОВ

Учебное пособие

Редактор, корректор М.В. Никулина

Компьютерная верстка М.А. Башкин, В.Г. Дурнев

Подписано в печать 15.07.08. Формат 60 × 84 1/8.

Бумага тип.

Усл. печ. л. 28,82. Уч.-изд. л. 12,0. Тираж 120 экз.

Заказ

Оригинал-макет подготовлен в редакционно-издательском отделе ЯрГУ

150000 Ярославль, ул. Советская, 14

Отпечатано ООО “Ремдер” ЛР ИД 06151 от 26.10.01

г. Ярославль, пр. Октября, 94, оф. 37, тел. (0852) 73–35–03