Федеральное государственное бюджетное учреждение науки
Институт математики им. С. Л. Соболева
Сибирского отделения Российской академии наук

На правах рукописи

Рыбалов Александр Николаевич

ГЕНЕРИЧЕСКИЙ ПОДХОД К АЛГОРИТМИЧЕСКИМ ПРОБЛЕМАМ

01.01.06 – математическая логика,
алгебра и теория чисел
ДИССЕРТАЦИЯ
на соискание учёной степени
доктора физико-математических наук

Научный консультант
д.ф.-м.н., профессор
В.Н. Ремесленников

Омск 2018

Содержание

Введение

Диссертация посвящена генерическому подходу к вычислимости и вычислительной сложности алгоритмических проблем — новому направлению исследований, возникшему на стыке теории вычислимости, теории сложности вычислений и комбинаторной алгебры.

Генерический подход был предложен в 2003 году И.Каповичем, А.Г.Мясниковым, В.Шпильрайном и П.Шуппом в работе [61]. В рамках этого подхода изучается поведение алгоритмов на множестве «почти всех» входов (это множество называется генерическим), игнорируя поведение алгоритма на остальных входах, на которых алгоритм может работать медленно или вообще не останавливаться. Такой подход имеет приложения в криптографии, где требуется, чтобы алгоритмические проблемы были трудными для «почти всех» входов. Понятие «почти все» формализуется введением асимптотической плотности на множестве входных данных I: для подмножества $S \subseteq I$ определяется последовательность

$$\rho_n(S) = \frac{|S_n|}{|I_n|},\ n = 1, 2, 3, \ldots,$$

где I_n – множество входов размера n, а $S_n = S \cap I_n$ – множество входов из S размера n. *Асимптотической плотностью* S называется предел (если он существует)

$$\rho(S) = \lim_{n\to\infty} \rho_n(S).$$

Множество S называется *генерическим*, если $\rho(S) = 1$, и *сильно генерическим*, если последовательность $\rho_n(S)$ экспоненциально быстро сходится к 1. С точки зрения практики, алгоритмы, решающие быстро проблему на генерическом множестве, так же хороши, как и быстрые алгоритмы для всех входов. Классическим примером такого алгоритма являет-

ся симплекс-метод: В. Кли и Г. Минти показали [65], что этот алгоритм имеет экспоненциальную сложность в худшем случае, но А. М. Вершик и П. В. Спорышев [22, 23] и, независимо от них, С. Смейл [80] доказали, что он за полиномиальное время решает задачу линейного программирования для большинства входных данных. В теории сложности вычислений поведение алгоритмов на множестве «почти всех» входов традиционно изучается в рамках подхода к сложности в среднем [47, 67, 33, 81], при этом время работы алгоритма усредняется по всему множеству входных данных. В отличие от сложности в среднем, генерический подход является более универсальным, так как может оказаться, что на множестве «плохих» входов даже усредненное время работы алгоритма неполиномиально. В то же время генерический алгоритм просто игнорирует эти входы. Более того, генерический подход применим также и к алгоритмически неразрешимым проблемам. Таким образом, может оказаться, что проблема трудноразрешима или вообще неразрешима в классическом смысле, но легкоразрешима в генерическом смысле. В работах А. Г. Мясникова, В. Н. Ремесленникова, А. В. Боровика, В. А. Романькова, П. Шуппа, Р. Гилмана, В. Дикерта, М. Камбитеса и др. [61, 34, 35, 40, 41, 46, 52, 60, 62, 71, 72, 73] было доказано, что таким поведением обладают многие алгоритмически неразрешимые проблемы алгебры и теории алгоритмов. Для многих классических NP-полных и трудноразрешимых проблем полиномиальные генерические алгоритмы были предложены в работах П. Эрдеша, Л. Бабаи, Р. Карпа, Ю. Гуревича, С. Шелаха, Э. Х. Гимади, Н. И. Глебова, В. А. Перепелицы и др. [4, 44, 48, 28, 63].

В тоже время, большой интерес как с теоретической точки зрения, так

и с точки зрения практических приложений, представляют алгоритмические проблемы, которые остаются неразрешимыми или трудноразрешимыми и в генерическом случае. Например, в современной криптографии интересны такие проблемы, которые, являясь (гипотетически) трудными в классическом смысле, остаются трудными и в генерическом смысле, т.е. для почти всех входов. Это объясняется тем, что при случайной генерации ключей в криптографическом алгоритме происходит генерация входа некоторой трудной алгоритмической проблемы, лежащей в основе алгоритма. Если проблема будет генерически легкоразрешимой, то для почти всех таких входов ее можно будет быстро решить и ключи почти всегда будут нестойкими. Поэтому проблема должна быть генерически трудной.

А. Г. Мясников и Дж. Хэмкинс в [52] доказали, что проблема остановки для машин Тьюринга с лентой, бесконечной в одну сторону, проблема остановки является генерически разрешимой. Там же ими была поставлена следующая проблема.

Проблема 1. *Является ли проблема остановки для машин Тьюринга с лентой, бесконечной в обе стороны, генерически разрешимой?*

В данной диссертации доказывается, что проблема остановки неразрешима на сильно генерических множествах входов. Также доказывается ее генерическая неразрешимость для так называемых нормализованных машин Тьюринга. В программе нормализованной машины Тьюринга из любого нефинального состояния q_i в соответствующих ему правилах в программе возможен переход в состояния q_j, где $j \leq mi + 1$, где m – размер рабочего алфавита. Смысл этого ограничения соответствует процессу экономного написания программы: начиная с первого правила, в

каждом очередном новом правиле можно либо переходить в одно из старых состояний, либо в одно новое. Это ограничение никак не уменьшает множество вычислимых функций. Полученные результаты верны для машин с лентой, бесконечной и в одну сторону, и в обе. Первый результат в частности говорит о том, что генерический алгоритм Мясникова-Хэмкинса нельзя сделать сильно генерическим. Из второго результата следует, что алгоритм Мясникова-Хэмкинса не является генерическим для нормализованных машин. В качестве приложения развитой техники, доказывается генерический аналог теоремы Клини о неподвижной точке [66, 1]. Отметим, что проблема остановки изучалась также и в рамках других подходов Г. Чейтиным, К. Калуде, А. Шенем и др. [31, 36, 37].

Классические примеры конечно определенных полугрупп с алгоритмически неразрешимой проблемой равенства слов были построены А. А. Марковым [10], Э. Постом [75], Г. С. Цейтиным [25], Ю. В. Матиясевичем [11]. Однако, как было показано М. Камбитесом [60], проблема равенства слов в этих полугруппах генерически разрешима за полиномиальное время. Причина этого явления состоит в том, что при интерпретации какой-либо неразрешимой проблемы (например, проблемы остановки для машин Тьюринга) внутри данной конечно определенной полугруппы, привносится много «мусора» в виде вспомогательных порождающих и определяющих соотношений для моделирования вычислений машин Тьюринга, что делает проблему равенства легкоразрешимой для большинства входов. Возникает следующая естественная проблема.

Проблема 2. *Существуют ли конечно определенные полугруппы с генерически неразрешимой проблемой равенства слов?*

В данной диссертации строятся примеры таких полугрупп. Остается

открытым вопрос о существовании конечно определенной группы с генерически неразрешимой проблемой равенства слов. Р. Гилман, А. Г. Мясников и Д. Осин [45] построили пример конечно определенной группы с проблемой равенства слов, неразрешимой на сильно генерических подмножествах. Используя неравенство Голода-Шафаревича, А. Г. Мясников и Д. Осин [69] построили конечно порожденную группу с рекурсивным множеством определяющих соотношений, проблема равенства слов в которой генерически неразрешима. А. Г. Мясников и Б. Хуссаинов в [64] построили пример рекурсивно определенной группы с проблемой равенства слов, неразрешимой на множествах ненулевой асимптотической плотности.

В 1970 году Ю. В. Матиясевич, основываясь на работах М. Девиса, Дж. Робинсон и Х. Патнема, доказал [12], что 10-я проблема Гильберта алгоритмически неразрешима, то есть не существует алгоритма, который по любому диофантовому уравнению определяет, разрешимо ли оно в целых числах В дальнейшем, Ю. В. Матиясевич и Дж. Робинсон в [68] показали, что десятая проблема Гильберта неразрешима для диофантовых уравнений с числом неизвестных $m \leq 13$. Д. Джонс в [56] снизил эту границу до 9. Б. Пунен и Д. Ф. Волох [74] изучили разрешимость «случайных» диофантовых уравнений над рациональными числами для некоторых естественных представлений. А. Г. Мясников и В. А. Романьков [70, 18, 19] показали, что основные функции шифрования многих криптографических систем с открытым ключом, среди которых система RSA и системы, основанные на трудноразрешимости проблемы дискретного логарифма, записываются на языке диофантовых уравнений. Эффективная генерическая разрешимость этих уравнений приводит к

взлому соответствующих систем, поэтому важной является следующая проблема.

Проблема 3. *Является ли проблема разрешимости диофантовых уравнений генерически (легко) разрешимой?*

В диссертации эта проблема изучается для двух представлений диофантовых уравнений: с помощью арифметических схем и с помощью систем Сколема [79].

Большой пласт алгоритмических проблем связан с элементарными теориями различных алгебраических систем – см. прекрасный обзор Ю. Л. Ершова, И. А. Лаврова, А. Д. Тайманова и М. А. Тайцлина [6]. Как правило, эти теории либо неразрешимы, либо имеют очень большую вычислительную сложность в худшем случае. М. Рабин и М. Фишер [49] доказали, что любой алгоритм, разрешающий теорию натуральных чисел с операцией сложения (так называемая арифметика Пресбургера), имеет как минимум дважды экспоненциальную сложность от длины формулы. Более точно: для любого алгоритма $\mathcal{A}$, распознающего арифметику Пресбургера, существует формула Φ длины n такая, что алгоритм $\mathcal{A}$ работает на формуле Φ за время, большее 2^{2^n}. К проблемам разрешимости элементарных теорий непосредственно примыкает знаменитая теорема Гёделя о неполноте [51, 3], которая утверждает, что если формальная арифметика непротиворечива, то она неполна, то есть в ней существует недоказуемое и неопровержимое утверждение. Возникает вопрос, а если ограничиться не всеми утверждениями, а «почти всеми», что можно сказать о такой генерической полноте формальной арифметики?

Проблема 4.

1. *Существуют ли неразрешимые элементарные теории, которые являются генерически разрешимыми?*

2. *Будет ли арифметика Пресбургера генерически разрешимой за полиномиальное время?*

3. *Существует ли генерическое множество замкнутых арифметических формул, для любой из которых либо сама формула, либо ее отрицание выводимо из аксиом формальной арифметики (при условии ее непротиворечивости)?*

В диссертации дается отрицательное решение всех пунктов этой проблемы для естественного представления формул первого порядка с помощью бинарных деревьев. Также в диссертации изучаются вопросы о генерической трудноразрешимости следующих классических алгоритмических проблем: проблема выполнимости булевых формул, проблемы дискретного логарифма, проблемы извлечения корня в группах вычетов, проблемы поиска изоморфизма графов, проблемы распознавания квадратичных вычетов. Для получения всех результатов о генерической неразрешимости и трудноразрешимости был предложен метод генерической амплификации.

В 2012 г. К. Джокуш и П. Шупп [55] начали изучение классической теории вычислимости в рамках генерического подхода. В частности, они ввели понятие грубой вычислимости, когда алгоритм может ошибаться на пренебрежимом множестве входов. Также они изучили связь между генеричностью и такими классическими понятиями теории вычислимости, как иммунность, би-иммунность, гипериммунность

и др. Кроме того, ими были введены аналоги тьюринговой сводимости для генерической вычислимости. Эта статья привлекла внимание специалистов по теории вычислимости к генерическому подходу и вызвала большое количество публикаций К. Джокуша, С. Лемпа, Р. Доуни, Д. Диамондстоуна, Д. Хиршфельда, Э. Астора, П. Чолака, Г. Игусы и др. [26, 27, 30, 31, 38, 42, 43, 53, 54, 57, 58], посвященных генерической теории вычислимости. Также в этой статье была поставлена следующая проблема.

Проблема 5. *Какова структура степеней (в том числе, рекурсивно перечислимых) генерической сводимости? В частности, существуют ли минимальные степени, минимальные пары для генерической сводимости?*

В работах П. Чолака и Г. Игусы [38, 57, 58] были изучены проблемы существования минимальных степеней (не обязательно рекурсивно перечислимых) и минимальных пар относительно генерической сводимости и было дано условное решение этой проблемы по модулю некоторого утверждения о структуре степеней генерической сводимости. В данной диссертации вводятся несколько новых типов генерических сводимостей. Изучается структура рекурсивно перечислимых степеней относительно этих сводимостей, в частности, доказывается несуществование минимальных рекурсивно перечислимых генерических степеней.

Важнейшим понятием классической теории сложности вычислений является понятие полиномиальной сводимости алгоритмических проблем. С его помощью можно сравнивать проблемы по вычислительной сложности и развивать богатую теорию NP-полноты [5]. Л. Левин [67] ввел понятие полиномиальной сводимости и NP-полноты в среднем.

Ю. Гуревич [47] построил примеры алгоритмических проблем, которые являются NP-полными в среднем. В данной диссертации вводится полиномиальная генерическая сводимость, определяются генерические классы P и NP, строятся примеры генерически NP-полных проблем.

Т. Бейкер, Дж. Гилл и Р. Соловей в 1975 г. доказали [29], что существуют два оракула A и B такие, что $\mathrm{P}^A = \mathrm{NP}^A$, но $\mathrm{P}^B \neq \mathrm{NP}^B$. Тем самым, они показали, что неравенство $\mathrm{P} \neq \mathrm{NP}$ не может быть доказано с использованием метода диагонализации: если с помощью диагонализации доказано неравенство каких-либо классов алгоритмических проблем $C_1 \neq C_2$, то для любого оракула A неравенство $C_1^A \neq C_2^A$ будет также верно (принцип релятивизации). В связи с этим представляет интерес следующая проблема.

Проблема 6. *Является ли генерический аналог проблемы о равенстве классов вычислительной сложности P и NP «проще» классической проблемы P vs NP? Можно ли использовать диагонализацию для доказательства неравенства генерических аналогов классов P и NP?*

В диссертации доказывается генерический аналог теоремы Бейкера-Гилла-Соловея, который, как и классическая теорема Бейкера-Гилла-Соловея, говорит о том, что «традиционнные» методы теории вычислимости, типа диагонализации, эту проблему не решают.

Основные результаты. На защиту выносятся следующие результаты (ниже в виде ссылок указаны работы, где был опубликован соответствующий результат).

1. Предложен метод генерической амплификации алгоритмических проблем [84]. С его помощью получены следующие результаты:

(a) Доказана генерическая неразрешимость проблемы остановки для машин Тьюринга [83, 95].

(b) Доказана генерическая неразрешимость десятой проблемы Гильберта [87, 88, 99].

(c) Доказана генерическая неразрешимость элементарных теорий, неразрешимых в классическом смысле [84, 86, 92].

(d) Построен пример полугруппы с генерически неразрешимой проблемой равенства слов [84].

(e) Доказано, что арифметика Пресбургера генерически неразрешима за экспоненциальное время [85].

(f) Доказано, что проблема выполнимости булевых формул генерически неразрешима за полиномиальное время при условии $\mathrm{P} \neq \mathrm{NP}$ и $\mathrm{P} = \mathrm{BPP}$ [100].

(g) Доказана генерическая трудноразрешимость проблемы распознавания квадратичных вычетов [90], проблемы дискретного логарифма [93], проблемы извлечения корня в группах вычетов [102], проблемы поиска изоморфизма графов [91], при условии трудноразрешимости этих проблем в худшем случае.

(h) Доказан генерический аналог теоремы Гёделя о неполноте формальной арифметики [89, 97].

(i) Доказан генерический аналог теоремы Клини о неподвижной точке [98].

2. Изучен вопрос о границах применимости метода генерической амплификации: построен пример генерически неразрешимой проблемы, которая не может быть получена с помощью генерической ам-

плификации из неразрешимой в худшем случае проблемы [104]; доказано, что любое рекурсивно перечислимое множество положительной асимптотической плотности может быть получено генерической амплификацией множества натуральных чисел [104]. Доказано существование абсолютно неразрешимых проблем, которые неразрешимы на любом множестве ненулевой асимптотической плотности [84].

3. Предложены генерические аналоги сводимостей алгоритмических проблем: gm-сводимость [105], генерическая клонирующая сводимость [94], генерическая тьюрингова сводимость [104]. Изучена структура рекурсивно перечислимых степеней относительно генерической тьюринговой сводимости: доказано существование полных степеней [104], доказано существование несравнимых степеней (аналог теоремы Мучника-Фридберга) [104], отсутствие минимальных и максимальных неполных степеней [104], доказан генерический аналог классической теоремы Сакса о разложении [104].

4. Предложен генерический аналог полиномиальной сводимости [96]. Доказана генерическая NP-полнота проблемы выполнимости булевых формул [96] и ограниченной проблемы остановки для машин Тьюринга [101].

5. Доказан генерический аналог теоремы Бейкера-Гилла-Соловея о релятивизации проблемы $\mathrm{P} \neq \mathrm{NP}$ [103].

Результаты диссертации докладывались на Омском алгебраическом семинаре (2006-2018), международных математических конференциях «Мальцевские чтения» (Новосибирск, 2006-2017), международных ма-

тематических конференциях «Methods of Logic in Mathematics» (Санкт-Петербург, 2006, 2008), международной конференции «Computer Science in Russia» (Екатеринбург, 2007), международной математической конференции «Leonard Euler and Modern Combinatorics» (Санкт-Петербург, 2007), посвященной 300-летию Л. Эйлера, конференции «Стохастические модели в биологии и предельные алгебры» (Омск, 2010), алгебраическом семинаре в Stevens Institute of Technology (Хобокен, США, 2005, 2007, 2011, 2016), конференции «Аппроксимация логических моделей, алгоритмов и задач» (Омск, 2015), конференции SIBECRYPT-15 (Новосибирск, 2015), конференции SIBECRYPT-17 (Красноярск, 2017), международной конференции «Математика в современном мире» (Новосибирск, 2017), на семинаре по дискретной математике лаборатории математической логики Санкт-Петербургского отделения Математического института им. В.А.Стеклова РАН (Санкт-Петербург, 2017), на Колмогоровском семинаре по сложности вычислений и сложности определений кафедры математической логики и теории алгоритмов механико-математического факультета МГУ им. М. В. Ломоносова (Москва, 2017), на международной алгебраической конференции памяти А. Г. Куроша (Москва, 2018), на конференции «Computability in Europe» (Киль, Германия, 2018).

Диссертация изложена на 171 странице, содержит введение, раздел с предварительными сведениями, четыре раздела с полученными результатами, заключение и список литературы. Разделы разбиты на подразделы, список литературы содержит 105 наименований. Нумерация утверждений (теорем, лемм, следствий), сквозная внутри каждого раздела и состоит из двух чисел: первое число — это номер раздела, второе — порядковый номер внутри раздела.

Результаты диссертации опубликованы в работах [83]– [105], входящих в перечень ВАК рецензируемых научных журналов, в которых должны быть опубликованы основные научные результаты диссертаций на соискание ученых степеней доктора и кандидата наук. Работа [84] написана в соавторстве с А.Г.Мясниковым. В диссертацию вошли результаты из [84], которые получены автором самостоятельно.

Используемые обозначения

$\mathbb{N} = \{1, 2, 3, \ldots\}$ – натуральные числа без нуля.

$\omega = \{0, 1, 2, 3, \ldots\}$ – натуральные числа с нулем.

$P(A)$ – множество всех подмножеств множества A.

$|A|$ – мощность конечного множества A.

$|w|$ – длина слова w над конечным алфавитом A.

A^* – множество всех конечных слов конечного алфавита A.

$\overline{A} = I \setminus A$ – дополнение множества $A \subseteq I$.

A_n – множество всех элементов размера n множества A.

$M(x)\downarrow$ – машина M останавливается на входе x.

Содержание работы В разделе 1 даются основные определения и факты теории алгоритмов, теории сложности вычислений и теории генерической вычислимости. Для подмножества $S \subseteq I$ определим последовательность

$$\rho_n(S) = \frac{|S_n|}{|I_n|},\ n = 1, 2, 3, \ldots,$$

где I_n – множество входов размера n, а $S_n = S \cap I_n$ – множество входов из S размера n. *Асимптотической плотностью* S назовем предел (если он существует)

$$\rho(S) = \lim_{n\to\infty} \rho_n(S).$$

Множество S называется *генерическим*, если $\rho(S) = 1$, *пренебрежимым*,

если $\rho(S) = 0$, *сильно пренебрежимым*, если последовательность $\rho_n(S)$ экспоненциально быстро сходится к 0 и *сильно генерическим*, если его дополнение $\overline{S}$ сильно пренебрежимо. Алгоритм $\mathcal{A} : I \to J \cup \{?\}$ называется *(сильно) генерическим*, если $\mathcal{A}$ останавливается на всех входах из I и множество $\{x \in I : \mathcal{A}(x) =?\}$ (сильно) пренебрежимо. Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если для всех $x \in I$ $\mathcal{A}(x) = y \in J \Rightarrow f(x) = y$. Множество $S \subseteq I$ и соответствующая проблема распознавания (S, I)

- *генерически разрешимы*, если существует генерический алгоритм, вычисляющий характеристическую функцию S.
- *сильно генерически разрешимы*, если существует сильно генерический алгоритм, вычисляющий характеристическую функцию S.

Множество $S \subseteq I$ и соответствующая проблема распознавания (S, I)

- *супер неразрешимы*, если S не является генерически разрешимым.
- *сильно неразрешимы*, если S не является сильно генерически разрешимым.

Отметим, что данное определение генерической разрешимости отличается от определения генерической разрешимости используемое К. Джокушем и П. Шуппом в [55], где для распознавания множества A применяются алгоритм $\mathcal{A}$ с генерическим множеством остановки $H_{\mathcal{A}}$: внутри $H_{\mathcal{A}}$ алгоритм корректно вычисляет характеристическую функцию A, а вне $H_{\mathcal{A}}$ алгоритм не останавливается. Это более широкое определение генерической разрешимости – см., например, теорему 2.22 и следствие 2.24 в [55]. В связи с этим, будет отличаться и определение

генерической тьюринговой сводимости, которое будет введено в разделе 4.3. Однако наше определение представляется более «эффективным», так как в нем генерические алгоритмы всегда останавливаются. Кроме того, все известные примеры конкретных генерических алгоритмов для различных алгоритмических проблем тоже всегда останавливаются. Более того, все результаты диссертации о генерической неразрешимости аналогично могут быть доказаны для определения Джокуша-Шуппа.

Раздел 2 посвящен доказательству генерической неразрешимости классических алгоритмически неразрешимых проблем с помощью метода генерической амплификации, который описывается в подразделе 2.1.

Пусть I, J – некоторые множества с определенными на них функциями размера. *Клонирование* множества I в множестве J – это функция $C : I \to P(J)$, из I во множество всех подмножеств $P(J)$ множества J такая, что $\forall x, y \in I$

- либо $C(x) \cap C(y) = \varnothing$,
- либо $C(x) \subseteq C(y)$ или $C(y) \subseteq C(x)$.

Для множества $S \subseteq I$ определим *клон* $C(S)$ как объединение всех клонов элементов из S:

$$C(S) = \bigcup_{x \in S} C(x).$$

Будем называть клонирование C из I в J *эффективным*, если существует всюду определенная вычислимая функция $E : I \times \omega \to J$ такая, что для любого $x \in I$

$$C(x) = \{E(x, 0), E(x, 1), \ldots, \}.$$

Таким образом, с помощью алгоритма E можно эффективно перечислять все элементы каждого клона. Клонирование $C : I \to P(J)$ назы-

вается *непренебрежимым* (соответственно, *не сильно пренебрежимым*), если для любого $x \in I$ множество $C(x)$ не является пренебрежимым (соответственно, сильно пренебрежимым) в J. Если $\mathcal{D} = (L, I)$ – проблема распознавания в I, то проблема распознавания $C(\mathcal{D}) = (C(L), J)$ в J называется *клоном* $\mathcal{D}$ в J относительно C.

Теорема 2.2. *Пусть I, J – множества с функциями размера и $C : I \to P(J)$ – эффективное клонирование. Тогда для любой неразрешимой проблемы распознавания $\mathcal{D} = (L, I)$ в I имеет место следующее:*

1. *Если C – непренебрежимое клонирование, то проблема $C(\mathcal{D})$ супер неразрешима в J.*

2. *Если C – не сильно пренебрежимое клонирование, то проблема $C(\mathcal{D})$ сильно неразрешима в J.*

В подразделе 2.2 доказывается генерическая неразрешимость проблемы остановки для машин Тьюринга. Обозначим множество всех машин Тьюринга через $\mathcal{M}$. Машина Тьюринга называется *нормализованной*, если из любого нефинального состояния q_i в соответствующих ему правилах в программе возможен переход в состояния q_j, где $j \leq |A|i+1$, где A – рабочий алфавит машины. Обозначим множество всех нормализованных машин Тьюринга через $\mathcal{NM}$. Рассмотрим теперь один из вариантов классической проблемы остановки. Зафиксируем рабочий алфавит $A = \{0, 1, \square\}$, где $\square$ – пустой символ. Определим следующее множество

$$HP_{\mathcal{M}} = \{M \in \mathcal{M} : M \text{ останавливается на входе } 0\}.$$

Аналогично определим

$$HP_{\mathcal{NM}} = \{M \in \mathcal{NM} : M \text{ останавливается на входе } 0\}.$$

Проблемы $(HP_{\mathcal{M}}, \mathcal{M})$ и $(HP_{\mathcal{NM}}, \mathcal{NM})$ в классическом смысле являются алгоритмически неразрешимыми.

Теорема 2.3. *Верно следующее:*

1. *Проблема $(HP_{\mathcal{M}}, \mathcal{M})$ сильно неразрешима.*

2. *Проблема $(HP_{\mathcal{NM}}, \mathcal{NM})$ супер неразрешима.*

Доказывается генерический аналог классической теоремы Клини о неподвижной точке.

Теорема 2.4. *Имеет место следующее:*

1. *Для любого генерического алгоритма $\mathcal{A} : \mathcal{NM} \to \mathcal{NM} \cup \{?\}$ существует такая машина Тьюринга M, что $\mathcal{A}(M) \neq ?$ и машина $\mathcal{A}(M)$ вычисляет ту же функцию, что и M.*

2. *Для любого сильно генерического алгоритма $\mathcal{A} : \mathcal{M} \to \mathcal{M} \cup \{?\}$ существует такая машина Тьюринга M, что $\mathcal{A}(M) \neq ?$ и машина $\mathcal{A}(M)$ вычисляет ту же функцию, что и M.*

В подразделе 2.3 строятся примеры полугрупп с генерически неразрешимой проблемой равенства слов. Пусть $\mathfrak{S} = \langle A|R\rangle$ — конечно определенная полугруппа с множеством порождающих $A = \{a_1, \ldots, a_n\}$ и множеством определяющих соотношений $R = \{r_1 = s_1, \ldots, r_k = s_k\}$. Для символа $x \notin A$ положим

$$\mathfrak{S}_x = \langle A, x|R, x = xa_1, \ldots, x = xa_n, x = xx\rangle.$$

Теорема 2.5. *Если проблема равенства слов в полугруппе $\mathfrak{S}$ неразрешима, то проблема равенства слов в полугруппе $\mathfrak{S}_x$ супер неразрешима.*

В подразделе 2.4 доказывается генерическая разрешимость десятой проблемы Гильберта для двух способов представления диофантовых уравнений.

Будем отождествлять диофантово уравнение $P(x_1, \ldots, x_m) = 0$ с арифметической схемой, представляющей полином P. *Арифметическая схема* от переменных $x_1, \ldots, x_m$ состоит из конечного числа промежуточных переменных $x_{m+1}, \ldots, x_{n+m}$ и присваиваний (одно присваивание для каждой промежуточной переменной) одного из следующих типов:

- $x_i = x_j * x_k$, где $j, k < i$ и $* \in \{+, -, \times\}$,
- $x_i = x_j + 1$, где $j < i$,
- $x_i = x_j - 1$, где $j < i$.

Переменные $x_1, \ldots, x_m$ называются *входными переменными* схемы. Последняя промежуточная переменная x_{m+n} называется *выходной переменной* схемы. Размер схемы – это число присваиваний n. Обозначим через $\mathcal{D}$ множество всех диофантовых уравнений, представленных с помощью арифметических схем. Определим множество

$$SOL(\mathcal{D}) = \{P \in \mathcal{D} : \exists x_1 \in \mathbb{Z} \ldots \exists x_m \in \mathbb{Z}\ P(x_1, \ldots, x_m) = 0\}.$$

Из неразрешимости десятой проблемы Гильберта следует, что проблема распознавания $(SOL(\mathcal{D}), \mathcal{D})$ неразрешима в классическом смысле.

Теорема 2.6. *Проблема* $(SOL(\mathcal{D}), \mathcal{D})$ *сильно неразрешима.*

Система диофантовых уравнений записана в *форме Сколема* [79, 13], если каждое уравнение в ней имеет один из следующих типов:

1. $x_i = x_j x_k$,
2. $x_i = x_j + x_k$,

3. $x_i = 1$.

Будем называть систему в форме Сколема *нормализованной*, если в k-м уравнении системы могут встречаться только переменные x_i, где $i \leq 3k$. Обозначим через $\mathcal{S}$ множество всех нормализованных систем диофантовых уравнений в форме Сколема. Определим множество

$$SOL_{\mathbb{N}}(\mathcal{S}) = \{S \in \mathcal{S} : \text{система } S \text{ разрешима над } \mathbb{N}\}.$$

Здесь множество натуральных чисел $\mathbb{N}$ не содержит 0. Из неразрешимости десятой проблемы Гильберта следует, что проблема распознавания $(SOL_{\mathbb{N}}(\mathcal{S}), \mathcal{D})$ неразрешима в классическом смысле.

Теорема 2.8. *Проблема разрешимости диофантовых уравнений в форме Сколема $(SOL_{\mathbb{N}}(\mathcal{S}), \mathcal{D})$ генерически разрешима за полиномиальное время.*

Определим теперь множество

$$SOL_{\mathbb{Z}}(\mathcal{S}) = \{S \in \mathcal{S} : \text{система } S \text{ разрешима над } \mathbb{Z}\}.$$

Теорема 2.9. *Проблема разрешимости нормализованных систем диофантовых уравнений в форме Сколема над множеством целых чисел $(SOL_{\mathbb{Z}}(\mathcal{S}), \mathcal{D})$ сильно неразрешима.*

В подразделе 2.5 доказывается генерическая неразрешимость элементарных теорий, которые неразрешимы в худшем случае. Зафиксируем конечную сигнатуру

$$\sigma = \{P_1^{(a_1)}, \ldots, P_k^{(a_k)}, f_1^{(b_1)}, \ldots, f_m^{(b_m)}, c_1, \ldots, c_l\},$$

где P_i предикаты, f_i функции и c_i константы. Положим

$$K = K_\sigma = \max_{i=1,\ldots,k, j=1,\ldots,m} \{a_i, b_j + 1, 2\}.$$

Пусть $\mathfrak{A} = \langle A, \sigma \rangle$ – алгебраическая система сигнатуры σ. Назовем замкнутую формулу Φ сигнатуры σ *простой атомарной* если она имеет следующий вид:

1) $x_j = f_i(x_{i_1}, \ldots, x_{i_s})$,

2) $P_i(x_{i_1}, \ldots, x_{i_r})$,

3) $x_i = c_j$.

Замкнутая формула Φ сигнатуры σ имеет *натуральную пренексную* форму, если она имеет вид:

$$\Phi = Q_1 x_1 \ldots Q_t x_t \varphi,$$

где $Q_i \in \{\forall, \exists\}$ – кванторы, φ бескванторная формула, полученная с помощью конъюнкций, дизъюнкций из простых атомарных формул или их отрицаний. Заметим, что любая замкнутая формула может быть приведена с помощью эквивалентных преобразований к натуральной пренексной форме. При этом размер формулы увеличивается не более чем линейно.

Пусть теперь φ – бескванторная формула, которая является булевой комбинацией простых атомарных формул и их отрицаний. Естественным образом можно сопоставить формуле φ бинарное дерево T_φ, которое представляет конструкцию φ из простых атомарных формул и их отрицаний с помощью конъюнкций и дизъюнкций. Внутренние вершины T_φ помечены символами $\vee$ и $\wedge$, а листья T_φ помечены простыми атомарными или их отрицаниями. Если T_φ имеет n листьев, то не более Kn переменных могут встретиться в T_φ, поэтому в дальнейшем будем полагать, что все переменные T_φ лежат в множестве $x_1, \ldots, x_{Kn}$. Пусть

$\Phi = Q_1 x_1 \dots Q_t x_t \varphi$ – формула в натуральной пренексной форме. *Представление* Φ состоит из бинарного дерева T_φ, которое кодирует бескванторную часть φ, и кванторной приставки $Q_1 x_1 \dots Q_t x_t$. Если T_φ имеет n листьев, то длина кванторной приставки не более Kn. Под размером $size(\Phi)$ формулы Φ будем понимать число n листьев дерева T_φ. Обозначим через $\mathcal{F}$ множество всех формул в натуральной пренексной форме.

Теорема 2.10. *Пусть* $\mathfrak{A} = \langle A, \sigma \rangle$ *– алгебраическая система конечной сигнатуры* σ *с неразрешимой элементарной теорией* $Th(\mathfrak{A})$. *Проблема распознавания* $(Th(\mathfrak{A}), \mathcal{F})$ *является сильно неразрешимой.*

Будем называть формулу $\Phi \in \mathcal{F}$ *нормализованной*, если в дереве T_φ, представляющем бескванторную часть φ формулы Φ, для любой переменной x_i, $i > q$ найдется переменная x_{i-1}, расположенная либо в том же листе дерева, либо в более левом. Другими словами, переменные в листьях дерева T_φ занумерованы слева направо. В представление нормализованной формулы размера n, помимо дерева T_φ и кванторной приставки $Q_1 Q_2 \dots Q_{Kn}$, будет входить еще перестановка $\pi_\Phi = (i_1, i_2, \dots, i_{Kn})$ индексов переменных $(1, 2, \dots, Kn)$, показывающая на какую переменную x_m навешан квантор Q_m. Обозначим через $\mathcal{NF}$ множество всех нормализованных формул сигнатуры σ.

Теорема 2.11. *Пусть* $\mathfrak{A} = \langle A, \sigma \rangle$ *– алгебраическая система конечной сигнатуры* σ *с неразрешимой элементарной теорией* $Th(\mathfrak{A})$. *Проблема распознавания* $(Th(\mathfrak{A}), \mathcal{NF})$ *является супер неразрешимой.*

В подразделе 2.6 доказывается генерический аналог теоремы Гёделя о неполноте формальной арифметики.

Теорема 2.12. *Пусть формальная арифметика непротиворечива. Имеет место следующее:*

1. *Не существует сильно генерического множества арифметических формул $\mathcal{G} \subseteq \mathcal{F}$ такого, что для любой формулы $\Phi \in \mathcal{G}$ либо Φ, либо $\neg\Phi$ выводится из аксиом формальной арифметики.*

2. *Не существует генерического множества арифметических формул $\mathcal{G} \subseteq \mathcal{NF}$ такого, что для любой формулы $\Phi \in \mathcal{G}$ либо Φ, либо $\neg\Phi$ выводится из аксиом формальной арифметики.*

В подразделе 2.7 изучаются ограничения метода генерической амплификации.

Теорема 2.13. *Пусть $A \subseteq \omega$ – простое пренебрежимое множество. Тогда*

1. *Множество A супер неразрешимо.*

2. *Не существует непренебрежимого клонирования $C : \omega \to P(\omega)$ такого, что $A = C(S)$ для некоторого множества $S \subseteq \omega$.*

С другой стороны, любое рекурсивно перечислимое множество натуральных чисел с ненулевой асимптотической плотностью можно всегда получить из множества ω с помощью генерической амплификации.

Теорема 2.14. *Пусть $A \subseteq \omega$ – любое рекурсивно перечислимое множество с ненулевой асимптотической плотностью. Тогда существует эффективное непренебрежимое клонирование $C : \omega \to P(\omega)$ такое, что $A = C(\omega)$.*

Строится пример абсолютно неразрешимой проблемы. Будем называть множество $A \subseteq \omega$ *абсолютно неразрешимым*, если не существует алгоритма $\mathcal{A} : \omega \to \{0, 1, ?\}$ такого, что

1. $\forall x \in \omega\ \mathcal{A}(x)\downarrow$,

2. множество $\{x \in \omega : \mathcal{A}(x) \neq ?\}$ непренебрежимо,

3. $\mathcal{A}(x) \neq ? \Rightarrow \mathcal{A}(x) = \chi_A(x)$, где $\chi_A(x)$ – характеристическая функция множества A.

Теорема 2.15. *Пусть $A \subseteq \omega$ – простое пренебрежимое множество. Тогда A абсолютно неразрешимо.*

В разделе 3 доказывается генерическая трудноразрешимость некоторых классических алгоритмических проблем.

В подразделе 3.1 доказывается генерическая трудноразрешимость арифметики Пресбургера – теории первого порядка алгебраической системы $\mathfrak{N} = \langle \mathbb{N}, \sigma_{PA} \rangle$ с сигнатурой $\sigma_{PA} = \{+, 1\}$.

Теорема 3.1. *Не существует сильно генерического экспоненциального алгоритма, решающего $(Th(\mathfrak{N}), \mathcal{F})$.*

В подразделе 3.2 доказывается генерическая трудноразрешимость проблемы выполнимости булевых формул при условии трудноразрешимости этой проблемы в худшем случае. Пусть φ – булева формула в базисе $\{\vee, \wedge, \neg\}$. Естественным образом можно сопоставить формуле φ бинарное дерево T_φ, которое представляет конструкцию φ из переменных и их отрицаний с помощью конъюнкций и дизъюнкций. Внутренние вершины T_φ помечены символами $\vee$ и $\wedge$, а листья T_φ помечены переменными или их отрицаниями. Обозначим через $\mathcal{BF}$ множество всех булевых формул.

Теорема 3.3. *Если $P \neq NP$ и $P = BPP$, то не существует сильно генерического полиномиального алгоритма, решающего проблему выполнимости на множестве $\mathcal{BF}$.*

В подразделах 3.3-3.6 доказывается генерическая трудноразрешимость классических проблем криптографии: проблемы распознавания квадратичных вычетов, проблемы дискретного логарифма, проблемы извлечения корня в группах вычетов, проблемы поиска изоморфизма графов, при условии трудноразрешимости этих проблем в худшем случае.

Пусть $\mathbb{Z}/(m)$ – мультипликативная группа вычетов по модулю $m \in \mathbb{N}$. *Квадратичным вычетом* в группе $\mathbb{Z}/(m)$ называется любой элемент x, для которого существует $y \in \mathbb{Z}/(m)$ такой, что $x = y^2$. Рассмотрим следующее множество входов:

$$RES = \{(m, x) \in \mathbb{N}^2 : m = pq, \text{ где } p, q - \text{простые числа}, x \in \mathbb{Z}/(m)\}.$$

Под *проблемой распознавания квадратичных вычетов* понимается проблема распознавания (QR, RES), где

$$QR = \{(m, x) \in RES : x - \text{ квадратичный вычет в } \mathbb{Z}/(m)\}.$$

Рассмотрим любую бесконечную последовательность натуральных чисел $\mu = \{m_1, m_2, \ldots\}$ такую, что для любого n имеет место $2^n < m_n < 2^{n+1}$ и m_n – произведение двух различных простых чисел. Будем называть такую последовательность *экспоненциальной*. Теперь определим алгоритмическую проблему $(QR(\mu), RES(\mu))$ как ограничение проблемы распознавания квадратичных вычетов (QR, RES) на следующее множество входных данных:

$$RES(\mu) = \{(m, x) : m \in \mu, x \in \mathbb{Z}/(m)\}.$$

Теорема 3.1. *Если для проблемы (QR, RES) не существует полиномиального вероятностного алгоритма, то существует экспоненциальная последовательность μ такая, что проблема $(QR(\mu), RES(\mu))$ не является генерически полиномиально разрешимой.*

Проблема дискретного логарифма состоит в вычислении функции dl : $I \to \mathbb{N}$, где I – это множество троек (a, p, g_p) таких, что p – простое число, g_p – фиксированный первообразный элемент в поле $GF(p)$ и $a \in GF(p)$, $a \neq 0$. Сама функция dl определяется следующим образом:

$$dl(a, p, g_p) = x \Leftrightarrow g_p^x = a \in GF(p).$$

Рассмотрим любую бесконечную последовательность простых чисел $\pi = \{p_1, p_2, \ldots, p_n, \ldots\}$, удовлетворяющую условию $2^n \leq p_n < 2^{n+1}$ для любого n. Будем называть такую последовательность *экспоненциальной*. Теперь определим функцию dl_π как ограничение функции dl на множество троек (a, p, g_p) таких, что $p \in \pi$.

Теорема 3.2. *Если для вычисления функции dl не существует полиномиального вероятностного алгоритма, то существует экспоненциальная последовательность π такая, что для вычисления функции dl_π не существует генерического полиномиального алгоритма.*

Проблема извлечения корня в группах вычетов состоит в вычислении функции $root$: $I \to \mathbb{N}$, где I – это множество троек (a, e, m) таких, что $m = pq$ и p, q – различные простые числа, $e < m$, $(\varphi(m), e) = 1$ и $a \in \mathbb{Z}/(m)$. Сама функция $root$ определяется следующим образом:

$$root(a, e, m) = x \Leftrightarrow x^e = a \in \mathbb{Z}/(m).$$

Рассмотрим любую бесконечную последовательность пар натуральных чисел $\mu = \{(e_1, m_1), (e_2, m_2), \ldots\}$ такую, что $2^n < m_n < 2^{n+1}$ для любого n, m_n – произведение двух различных простых чисел, для любого $n > 1$, и $e_n < m_n$ и $(\varphi(m_n), e_n) = 1$ для любого n. Будем называть такую последовательность *экспоненциальной*. Теперь определим функцию $root_\mu$ как

ограничение функции *root* на следующее множество входных данных:

$$I = \{(a, e, m) : (e, m) \in \mu, a \in \mathbb{Z}/(m)\}.$$

Теорема 3.3. *Если для вычисления функции root не существует полиномиального вероятностного алгоритма, то существует экспоненциальная последовательность μ такая, что для вычисления функции $root_\mu$ не существует генерического полиномиального алгоритма.*

Напомним, что два графа G_1 и G_2 называются изоморфными, если существует биекция π между множествами вершин G_1 и G_2 такая, что для любых вершин v, u графа G_1 v и u соединены ребром в G_1 тогда и только тогда, когда $\pi(v)$ и $\pi(u)$ соединены ребром в G_2. Биекция π, осуществляющая изоморфизм, является перестановкой множества вершин графов G_1, G_2, если вершины обоих графов занумерованы числами $\{1, 2, \ldots, n\}$. Множеством входов для проблемы поиска изоморфизма графов является множество всех пар изоморфных графов. В этой проблеме требуется вычислять следующую функцию.

$$sgi(G_1, G_2) = \text{перестановка, осуществляющая изоморфизм } \pi$$

$$\text{между изоморфными графами } G_1 \text{ и } G_2.$$

Рассмотрим бесконечную последовательность графов $\gamma = \{G_1, G_2, \ldots, G_n, \ldots\}$ такую, что G_n имеет n вершин для любого n. Для каждой последовательности графов γ определим функцию sgi_γ – ограничение функции sgi на множество пар (G, G_n), где $G_n \in \gamma$ и граф G изоморфен графу G_n.

Теорема 3.4. *Если для вычисления функции sgi не существует полиномиального вероятностного алгоритма, то существует последо-*

вательность графов γ такая, что для вычисления функции sgi_γ не существует генерического полиномиального алгоритма.

В разделе 4 вводятся три типа генерических сводимостей. Изучаются их свойства. Изучается структура рекурсивно перечислимых степеней самой общей — генерической тьюринговой сводимости.

Множество $A \subseteq \omega$ *генерически m-сводится* к $B \subseteq \omega$ (обозначается $A \leq_{gm} B$), если существует вычислимая функция $f : \omega \to \omega$ такая, что

1. $\forall x \in \omega \; x \in A \Leftrightarrow f(x) \in B$,
2. $\forall S \subseteq \omega \; S$ непренебрежимо $\Rightarrow f(S)$ непренебрежимо.

Множество $A \subseteq \omega$ *генерически клонированно сводится* к множеству $B \subseteq \omega$ (обозначается это $A \leq_{gc} B$), если либо $A = B$, либо существует такая всюду определенная вычислимая функция $f : \omega \times \omega \to \omega \cup \{?\}$, что

1. $\forall x \in \omega$ если $f(x,0) \neq ?$, то $\forall n \in \omega \; f(x,n) \neq ?$.
2. Множество $\{x \in \omega : f(x,0) = ?\}$ пренебрежимо.
3. $\forall x \in \omega$ если $f(x,0) \neq ?$, то множество $\{f(x,n) : n \in \omega\}$ непренебрежимо.
4. $\forall x \in \omega$ если $f(x,0) \neq ?$, то
   - $x \in A \Rightarrow \forall n \in \omega \; f(x,n) \in B$.
   - $x \notin A \Rightarrow \forall n \in \omega \; f(x,n) \notin B$.

Пусть A – произвольное множество натуральных чисел. *Генерическим оракулом* множества A называется функция $\varphi_A : \omega \to \{0,1,?\}$ такая, что

1. Множество $\{x : \varphi_A(x) = ?\}$ пренебрежимо.

2. $\forall x \in \omega\ \varphi_A(x) = 1 \Rightarrow x \in A$.

3. $\forall x \in \omega\ \varphi_A(x) = 0 \Rightarrow x \notin A$.

Множество $A \subseteq \omega$ *генерически сводится по Тьюрингу* к множеству $B \subseteq \omega$, если существует машина M с командами обращения к оракулу такая, что для любого генерического оракула φ_B генерический алгоритм M^{φ_B} вычисляет характеристическую функцию A. Обозначается это $A \leq_{gT} B$.

Теорема 4.10. *Пусть* $A, B \subseteq \omega$.

1. *Если* $A \leq_{gm} B$, *то* $A \leq_{gT} B$.

2. *Если* $A \leq_{gc} B$, *то* $A \leq_{gT} B$.

Будем писать $A \equiv_{gT} B$, если $A \leq_{gT} B$ и $B \leq_{gT} A$. Определим также генерическую тьюрингову степень множества A как

$$d_{gT}(A) = \{B \subseteq \omega : B \equiv_{gT} A\}.$$

Генерическая тьюрингова $\mathbf{a}$ степень рекурсивно перечислима, если содержит хотя бы одно рекурсивно перечислимое множество. Будем писать, что $\mathbf{a} \leq \mathbf{b}$, если существуют $A \in \mathbf{a}$ и $B \in \mathbf{a}$ такие, что $A \leq_{gT} B$. Аналогично определяется отношение $\mathbf{a} < \mathbf{b}$. Все генерически разрешимые множества образуют одну генерическую тьюрингову степень, которая обозначается $\mathbf{0}$. Для любой генерической тьюринговой степени $\mathbf{a}$ имеет место $\mathbf{0} \leq \mathbf{a}$.

Рекурсивно перечислимое множество A будем называть *gT-полным*, если для любого рекурсивно перечислимого B имеет место $B \leq_{gT} A$. Соответствующая степень называется *gT-полной*.

Теорема 4.4. *Существуют gT-полные генерические рекурсивно перечислимые степени.*

Следующее утверждение является аналогом классической теоремы Мучника-Фридберга о существовании несравнимых относительно тьюринговой сводимости рекурсивно перечислимых степеней.

Теорема 4.5. *Существуют несравнимые генерические тьюринговы рекурсивно перечислимые степени.*

Будем называть генерическую тьюрингову рекурсивно перечислимую степень $\mathbf{a}$ *максимальной*, если не существует неполной генерической тьюринговой рекурсивно перечислимой степени $\mathbf{b}$ такой, что $\mathbf{a} < \mathbf{b}$.

Теорема 4.6. *Не существует максимальной генерической рекурсивно перечислимой степени.*

Ненулевая генерическая тьюрингова рекурсивно перечислимая степень $\mathbf{a}$ называется *минимальной*, если не существует такой генерической тьюринговой рекурсивно перечислимой степени $\mathbf{b}$ что $\mathbf{0} < \mathbf{b} < \mathbf{a}$.

Теорема 4.7. *Не существует минимальной генерической тьюринговой рекурсивно перечислимой степени.*

Доказан генерический аналог классической теоремы Сакса о разложении.

Теорема 4.3. *Пусть A – супер неразрешимое рекурсивно перечислимое множество. Тогда $A = B_0 \cup B_1$, где B_0, B_1 – непересекающиеся рекурсивно перечислимые множества такие, что $A \not\leq_{gT} B_0$ и $A \not\leq_{gT} B_1$.*

В разделе 5 вводится понятие генерической полиномиальной сводимости алгоритмических проблем. Определяются генерические аналоги классов P и NP. Доказывается генерическая генерическая NP-полнота проблемы выполнимости булевых формул и ограниченной проблемы остановки для машин Тьюринга.

Пусть I, J — некоторые множества входов с определенными на них функциями размера. Множество $A \subseteq I$ *генерически полиномиально сводится* к множеству $B \subseteq J$ (обозначается $A \leq_{GenP} B$), если существуют вероятностный полиномиальный алгоритм $\mathcal{R} : I \times \mathbb{N} \to P(J) \cup \{?, !\}$, полином $p(n)$, полином $q(n)$ степени больше 2 и константа $C > 0$, такие, что

1. для всех $x \in I$ либо $\forall n\, \mathcal{R}(x, n) = \{?\}$, либо для всех $n \geq q(k)$, где $k = size(x)$, выполнены следующие условия:
   (a) $\forall y \in \mathcal{R}(x, n)\ (y \neq !\ \Rightarrow\ size(y) = n)$;
   (b) все элементы в $\mathcal{R}(x, n) \setminus \{!\}$ выдаются алгоритмом $\mathcal{R}$ равновероятно;
   (c) вероятность получить ответ «!» в $\mathcal{R}(x, n)$ не больше 2^{-Ck};
   (d) $\dfrac{|\mathcal{R}(x, n)|}{|J_n|} > \dfrac{1}{(p(n))^k}$;
   (e) $x \in A \Rightarrow \mathcal{R}(x, n) \subseteq B$;
   (f) $x \notin A \Rightarrow \mathcal{R}(x, n) \subseteq J \setminus B$;
2. множество $\{x \in I : \forall n\ (\mathcal{R}(x, n) = ?)\}$ сильно пренебрежимо.

Определим сильно генерический аналог класса NP. Пусть I – множество входов с определенной на нем функцией размера входа. Множество $S \subseteq I$ принадлежит *классу* sgNP, если существует полиномиальное сильно генерическое множество $G \subseteq I$, такое, что $S \cap G \in$ NP. Множество $S \in$ sgNP называется *генерически NP-полным*, если для любого $A \in$ sgNP имеет место $A \leq_{GenP} S$.

Теорема 5.3. *Проблема выполнимости булевых формул генерически NP-полна.*

Рассмотрим множество $BHP \subseteq \mathcal{M}$ машин Тьюринга M над алфовитом $\{0,1,\square\}$ таких, что существует $x \in \{0,1\}^*$ такой, что $|x| < size(M)$ и M останавливается на x за $\leq size(M)$ шагов и выдает 1.

Теорема 5.4. *Множество BHP генерически NP-полное.*

Пусть I – множество входов с определенной на нем функцией размера входа. Множество $S \subseteq I$ принадлежит *классу* genP, если существует полиномиальный генерический алгоритм, вычисляющий характеристическую функцию множества S. Множество $S \subseteq I$ принадлежит *классу* genNP, если существует разрешимое за полиномиальное время генерическое множество $G \subseteq I$ такое, что $S \cap G \in$ NP. Аналогично определяются релятивизованные генерические классы genP и genNP.

Доказывается генерический аналог теоремы Бейкера-Гилла-Соловея о релятивизации проблемы P $\neq$ NP.

Теорема 5.5. *Существуют такие оракулы A и B, что $genP^A = genNP^A$ и $genP^B \neq genNP^B$.*

Благодарности.

Автор выражает благодарность своему научному консультанту Владимиру Никаноровичу Ремесленникову, а также профессорам Алексею Георгиевичу Мясникову и Виталию Анатольевичу Романькову.

1 Предварительные сведения

1.1 Предварительные сведения из теории алгоритмов

В этом подразделе приводятся основные определения и факты теории алгоритмов, которые будут в дальнейшем использоваться. Их доказательства можно найти в многочисленных монографиях и учебных пособиях [2, 16, 9, 20].

Будем рассматривать одноленточные машины Тьюринга с рабочим алфавитом $A = \{a_1, \ldots, a_m\}$. Программа P_M машины Тьюринга M с множеством внутренних состояний $Q = \{q_0, q_1, \ldots, q_n\}$ над алфавитом A имеет вид

$$
\begin{aligned}
&(q_1, a_1) \to (q_{j_1}, b_1, S_1),\\
&(q_1, a_2) \to (q_{j_2}, b_2, S_2),\\
&\ldots\\
&(q_1, a_m) \to (q_{j_m}, b_m, S_m),\\
&(q_2, a_1) \to (q_{j_{m+1}}, b_{m+1}, S_{m+1}),\\
&\ldots\\
&(q_n, a_m) \to (q_{j_{nm}}, t_{nm}, S_{nm}).
\end{aligned}
$$

Программа содержит по одному правилу $(q_i, a) \to (q_j, b, S)$ для всех ненулевых (нефинальных) состояний q_i и символов алфавита $a \in A$. Всего получается $n|A|$ правил. Каждое такое правило предписывает, что машина, находясь в нефинальном состоянии q_i и видя кареткой на ленте символ $a \in A$, должна перейти в состояние q_j (оно может быть и финальным q_0), записать на ленте символ $b \in A$ и сдвинуть каретку согласно значению сдвига $S \in \{R, L\}$ (вправо, влево).

Машина Тьюринга M над алфавитом $A \cup \{\square\}$ *вычисляет* функцию $f_M : A^* \to A^*$, если для любых $w_1, w_2 \in A^*$ имеет место $f_M(w_1) = w_2$

$\Leftrightarrow$ M начиная работу с записанным на ленте словом w_1 (в остальных ячейках ленты – пустой символ $\square$), с кареткой, находящейся над первым символом слова w_1, работает согласно правилам программы P_M, останавливается, и после остановки на ленте записано слово w_2. Также машина M не останавливается на $w_1 \Leftrightarrow$ значение f_M не определено на w_1.

Характеристической функцией множества $A \subseteq I$ называется функция $\chi_A : I \to \{0, 1\}$, определенная следующим образом:

$$\chi_A(x) = \begin{cases} 1, \text{ если } x \in A, \\ 0, \text{ если } x \notin A. \end{cases}$$

Множество $A \subseteq \omega$ называется *рекурсивным*, если его характеристическая функция χ_A вычислима.

Частичной характеристической функцией множества $A \subseteq I$ называется функция

$$\hat{\chi}_A(x) = \begin{cases} 1, \text{ если } x \in A, \\ \text{не определено, если } x \notin A. \end{cases}$$

Множество $A \subseteq \omega$ называется *рекурсивно перечислимым*, если его частичная характеристическая функция $\hat{\chi}_A$ вычислима. Нам понадобится следующая теорема о рекурсивно перечислимых множествах.

Теорема 1.1. *Пусть $A \subseteq \omega$ – бесконечное рекурсивно перечислимое множество. Тогда существует инъективная вычислимая всюду определенная функция $f : \omega \to \omega$ такая, что $A = \{f(0), f(1), \ldots, f(n), \ldots\}$.*

Другими словами, функция f перечисляет множество A без повторений. Бесконечный набор $\{A_s, s \in \omega\}$ конечных начальных участков

$$A_s = \{f(0), f(1), \ldots, f(s)\}$$

будем называть *эффективным перечислением* рекурсивно перечислимого множества A.

Множество $A \subseteq \omega$ называется *иммунным*, если оно бесконечно и не содержит бесконечных рекурсивно перечислимых подмножеств. Множество $B \subseteq \omega$ называется *простым*, если B рекурсивно перечислимо и $\overline{B}$ иммунно.

Множество $A \subseteq \omega$ *m-сводится* к множеству $B \subseteq \omega$ (обозначается $A \leq_m B$), если существует вычислимая всюду определенная функция $f : \omega \to \omega$ такая, что

$$\forall x \in \omega \; x \in A \Leftrightarrow f(x) \in B.$$

Множество $A \subseteq \omega$ *m-эквивалентно* множеству $B \subseteq \omega$ (обозначается $A \equiv_m B$), если $A \leq_m B$ и $B \leq_m A$. Через $A <_m B$ обозначается тот факт, что $A \leq_m B$ и $B \not\leq_m A$. *m-степень* множества $A \subseteq \omega$ определяется как

$$d_m(A) = \{B \subseteq \omega : B \equiv_m A\}.$$

Множество $A \subseteq \omega$ *1-сводится* к множеству $B \subseteq \omega$ (обозначается $A \leq_1 B$), если существует инъективная вычислимая всюду определенная функция $f : \omega \to \omega$ такая, что

$$\forall x \in \omega \; x \in A \Leftrightarrow f(x) \in B.$$

1-эквивалентность, 1-степени определяются аналогично m-сводимости. Нам понадобится следующая теорема об 1-сводимости рекурсивных множеств.

Теорема 1.2. *Пусть $A, B \subseteq \omega$ – бесконечные ко-бесконечные рекурсивные множества. Тогда $A \equiv_1 B$.*

Машина с оракулом $A \subseteq \omega$ в своей программе может содержать команды обращения к оракулу $x \in A?$, которая отвечает на вопрос о принадлежности x множеству A. Множество $A \subseteq \omega$ *сводится по Тьюрингу* к множеству $B \subseteq \omega$, если существует машина M с командами обращения к оракулу B, которая вычисляет характеристическую функцию A. Обозначается это $A \leq_T B$. T-эквивалентность, тьюринговы степени определяются аналогично m-сводимости. Тьюрингова степень называется рекурсивно перечислимой, если она содержит рекурсивно перечислимое множество.

Сформулируем некоторые классические теоремы о структуре рекурсивно перечислимых степеней тьюринговой сводимости.

Теорема 1.3 (Мучник-Фридберг [15, 50])**.** *Существуют рекурсивно перечислимые множества* $A, B \subseteq \omega$ *такие, что* $A \not\leq_T B$ *и* $B \not\leq_T A$.

Теорема 1.4 (Теорема о разложении Сакса [77])**.** *Пусть* A *– неразрешимое рекурсивно перечислимое множество. Тогда* $A = A_0 \cup A_1$*, где* A_0, A_1 *– непересекающиеся рекурсивно перечислимые множества такие, что* $A \not\leq_T A_0$ *и* $A \not\leq_T A_1$.

Теорема 1.5 (Теорема о плотности Сакса [78])**.** *Для любых рекурсивно перечислимых множеств* $A, B \subseteq \omega$ *таких, что* $A <_T B$ *существует рекурсивно перечислимое множество* $C \subseteq \omega$ *такое, что* $A <_T C <_T B$.

Теорема 1.6. *Существует T-полное рекурсивно перечислимое множество* $A \subseteq \omega$*. То есть такое, что* $B \leq_T A$ *для любого рекурсивно перечислимого* $B \subseteq \omega$.

1.2 Предварительные сведения из теории сложности вычислений

В этом подразделе приведем некоторые сведения из теории сложности вычислений [5, 24].

Время работы $t_M(x)$ машины Тьюринга M над алфавитом A на входе $x \in A^*$ – это число шагов машины от начала работы до остановки. Если M на x не останавливается, полагаем $t_M(x) = \infty$.

Машина Тьюринга M *полиномиальна*, если существует полином $p(n)$ такой, что для любого $x \in A^*$ имеет место $t_M(x) < p(|x|)$. Под *эффективной нумерацией всех полиномиальных машин Тьюринга* будем подразумевать эффективную нумерацию всевозможных пар $\{(M_i, p_k(n))\}_{i\in\omega,k\in\omega}$, где M_i – машина Тьюринга с номером i, а $p_k(n) = n^k + k$. Такая пара на входе x моделирует работу некоторой полиномиальной машины Тьюринга следующим образом:

$$(M_i, p_k(n))(x) = \begin{cases} M_i(x), \text{ если } M_i(x)\downarrow \text{ за } \leq p_k(|x|) \text{ шагов,} \\ 0, \text{ иначе.} \end{cases}$$

Легко видеть, что любая полиномиальная машина Тьюринга встретится в этой последовательности.

Множество $S \subseteq I$ принадлежит *классу* P, если существует полиномиальная машина Тьюринга, вычисляющая характеристическую функцию множества S. Множество $S \subseteq I$ принадлежит *классу* NP, если существует такое множество $S' \in$ P и такой полином $p(n)$, что

$$x \in S \Leftrightarrow \exists y \in \{0,1\}^* \ : \ |y| < p(|x|) \text{ и } (x,y) \in S'.$$

Здесь строка y называется подсказкой (или сертификатом, или решением), а множество S' проверятелем.

Теперь определим релятивизованные версии этих классов. Пусть A есть некоторое подмножество I. *Машина с оракулом* A в своей программе может содержать команды обращения к оракулу, причем выполнение этой команды происходит за один шаг работы вычислительного устройства. Множество $S \subseteq I$ принадлежит *классу* P^A, если существует полиномиальная машина с оракулом A, вычисляющая характеристическую функцию множества S. Множество $S \subseteq I$ принадлежит *классу* NP^A, если существует если существует такое множество $S' \in \mathrm{P}^A$ и такой полином $p(n)$, что

$$x \in S \Leftrightarrow \exists y \in \{0,1\}^* \ : \ |y| < p(|x|) \text{ и } (x,y) \in S'.$$

Т. Бейкер, Дж. Гилл и Р. Соловей в 1975 г. доказали [29] следующую теорему.

Теорема 1.7 (Бейкер, Гилл, Соловей)**.** *Существуют два оракула* A *и* B *такие, что* $P^A = NP^A$*, но* $P^B \neq NP^B$*.*

Тем самым, они показали, что неравенство $\mathrm{P} \neq \mathrm{NP}$ не может быть доказано с использованием метода диагонализации: если с помощью диагонализации доказано неравенство каких-либо классов алгоритмических проблем $C_1 \neq C_2$, то для любого оракула A неравенство $C_1^A \neq C_2^A$ будет верно (принцип релятивизации).

Множество $A \subseteq I$ *полиномиально сводится* к множеству $B \subseteq J$, если существует функция $f : I \to J$, вычислимая на полиномиальной машине Тьюринга, такая, что

$$\forall x \in I \ x \in A \Leftrightarrow f(x) \in B.$$

Множество $A \in \mathrm{NP}$ называется *NP-полным*, если любое множество $B \in \mathrm{NP}$ полиномиально сводится к A. Соответствующая проблема рас-

познавания NP-полного множества также называется *NP-полной*. NP-полные проблемы являются самыми сложными проблемами в классе NP в том смысле, что если хотя бы одна из них лежит в классе P, то P = NP, а также, если P $\neq$ NP, то любая NP-полная проблема не лежит в классе P.

Булева формула $\varphi(x_1, \ldots, x_n)$ называется *выполнимой*, если существуют значения булевых переменных $x_1, \ldots, x_n$ при которых $\varphi(x_1, \ldots, x_n)$ является истинной. *Проблема выполнимости булевых формул* – это проблема распознавания множества выполнимых булевых формул на множестве всех булевых формул. В 1971 г. Кук доказал [39] следующую теорему.

Теорема 1.8 (Кук). *Проблема выполнимости булевых формул является NP-полной.*

Вероятностная машина Тьюринга – это машина Тьюринга, в программе которой допускаются пары правил вида

$$(q_i, a) \to (q_j, b, S_1)$$

$$(q_i, a) \to (q_k, c, S_2)$$

В процессе работы такой машины с вероятностью $1/2$ выбирается первое правило и с вероятностью $1/2$ второе. Обозначим

$$P(M(x) = y) = \text{вероятность того, что } M \text{ на входе } x \text{ выдает } y.$$

Время работы $t_M(x, \tau)$ вероятностной машины Тьюринга на входе x зависит от вычислительного пути (последовательности выполненных команд) τ. Проблема $S \subseteq I$ принадлежит *классу* BPP если существует вероятностная машина Тьюринга M и полином $p(n)$ такие, что

1. Для любого x и для любого вычислительного пути τ машины M на x имеет место $t_M(x, \tau) < p(|x|)$.

2. Если $x \in S$, то $P(M(x) = 1) > 2/3$.

3. Если $x \notin S$, то $P(M(x) = 0) > 2/3$.

Вероятностные машины Тьюринга формализуют понятие алгоритма, использующего генератор случайных чисел. Класс BPP – это класс проблем, эффективно решаемых такими вероятностными алгоритмами. Большинство исследователей сейчас считает, что имеет место равенство P = BPP. Это равенство означает, что любой полиномиальный вероятностный алгоритм можно эффективно дерандомизировать, т.е. построить полиномиальный детерминированный алгоритм, решающий ту же задачу. Хотя это равенство пока еще не доказано, имеются серьезные результаты в пользу него [59].

1.3 Предварительные сведения из теории генерической вычислимости

Пусть I – некоторое множество. Функция $size : I \to \mathbb{N}$ называется *функцией размера*, если для любого $n \in \mathbb{N}$ множество

$$I_n = \{x \in I : size(x) = n\}$$

конечно. Например, если $I = A^*$ – множество слов над конечным алфавитом A, то функцией размера будет функция $size(w) = |w|$ для любого $w \in A^*$. Также для множества натуральных чисел $\mathbb{N}$ функция размера сопоставляет любому натуральному числу длину его двоичной записи. Для подмножества $S \subseteq I$ определим последовательность

$$\rho_n(S) = \frac{|S_n|}{|I_n|},\ n = 1, 2, 3, \ldots,$$

где I_n – множество входов размера n, а $S_n = S \cap I_n$ – множество входов из S размера n. Заметим, что $\rho_n(S)$ это вероятность попасть в S при случайной и равновероятной генерации входов из I_n. *Асимптотической плотностью* S назовем предел (если он существует)

$$\rho(S) = \lim_{n\to\infty} \rho_n(S).$$

Множество S называется *генерическим*, если $\rho(S) = 1$ и *пренебрежимым*, если $\rho(S) = 0$. Очевидно, что S генерическое тогда и только тогда, когда его дополнение $I \setminus S$ пренебрежимо. Назовем множество S *сильно пренебрежимым*, если последовательность $\rho_n(S)$ экспоненциально быстро сходится к 0, т. е. существуют константы σ, $0 < \sigma < 1$, и $C > 0$, такие, что для любого n

$$\rho_n(S) < C\sigma^n.$$

Теперь S называется *сильно генерическим*, если его дополнение $\overline{S}$ сильно пренебрежимо.

Алгоритм $\mathcal{A}$ с множеством входов I и множеством выходов $J \cup \{?\}$ $(? \notin J)$ называется *(сильно) генерическим*, если

1. $\mathcal{A}$ останавливается на всех входах из I,
2. множество $\{x \in I : \mathcal{A}(x) =?\}$ (сильно) пренебрежимо.

Генерический алгоритм $\mathcal{A}$ вычисляет функцию $f : I \to J$, если для всех $x \in I$ $\mathcal{A}(x) = y \in J \Rightarrow f(x) = y$. Ситуация $\mathcal{A}(x) =?$ означает, что $\mathcal{A}$ не может вычислить функцию f на аргументе x. Но условие 2 гарантирует то, что $\mathcal{A}$ корректно вычисляет f на почти всех входах (входах из генерического множества). Множество $S \subseteq I$ и соответствующая проблема распознавания (S, I)

- *генерически разрешимы*, если существует генерический алгоритм, вычисляющий характеристическую функцию S.
- *сильно генерически разрешимы*, если существует сильно генерический алгоритм, вычисляющий характеристическую функцию S.

Множество $S \subseteq I$ и соответствующая проблема распознавания (S, I)

- *супер неразрешимы*, если S не является генерически разрешимым.
- *сильно неразрешимы*, если S не является сильно генерически разрешимым.

Отметим, что данное определение генерической разрешимости отличается от определения генерической разрешимости используемое в [55], где для распознавания множества A применяются алгоритм $\mathcal{A}$ с генерическим множеством остановки $H_{\mathcal{A}}$: внутри $H_{\mathcal{A}}$ алгоритм корректно вычисляет характеристическую функцию A, а вне $H_{\mathcal{A}}$ алгоритм не останавливается. Это более широкое определение генерической разрешимости – см., например, теорему 2.22 и следствие 2.24 в [55]. Однако наше определение представляется более «эффективным», так как в нем генерические алгоритмы всегда останавливаются. Кроме того, все известные примеры конкретных генерических алгоритмов для различных алгоритмических проблем тоже всегда останавливаются. В связи с этим, будет отличаться и определение генерической тьюринговой сводимости, которое будет введено в разделе 4.3.

Имеется существенное различие между генерически разрешимыми проблемами и сильно генерически разрешимыми проблемами. Допустим, имеется проблема S, разрешимая на некотором разрешимом за полино-

миальное время генерическом множестве G, для которого

$$\frac{|G \cap I_n|}{|I_n|} = \frac{n-1}{n}.$$

Таким образом G – генерическое, но не сильно генерическое множество. Теперь хоть и проблема S разрешима для почти всех входов, тем не менее, есть эффективный способ получить «плохой» вход, на котором генерический алгоритм не работает. Полиномиальный алгоритм для генерации плохих входов следующий.

1. Сгенерировать равномерно случайный вход x размера n.

2. Если $x \in G$, повторить шаг 1, иначе закончить.

Действительно, вероятность получить только хорошие входы за n^2 раундов:

$$\left(\frac{n-1}{n}\right)^{n^2} = \left(\left(1-\frac{1}{n}\right)^{n}\right)^{n} \to e^{-n}.$$

Поэтому с вероятностью, очень близкой к 1, будет получен плохой вход. С другой стороны, легко видеть, что если проблема разрешима на сильно генерическом множестве, то такой простой алгоритм генерации потребует экспоненциального числа раундов и будет неэффективным. Для приложений к криптографии, это означает, что просто генерическая легкоразрешимость проблемы не делает эту проблему бесполезной для создания на ее основе криптосистемы, так как для нее существует эффективная процедура генерации трудных входов. В то же время, сильно генерически легкоразрешимые проблемы в этом смысле бесполезны для криптографии.

2 Генерически неразрешимые алгоритмические проблемы

2.1 Генерическая амплификация алгоритмических проблем

Генерическая амплификация – это метод, который позволяет из алгоритмически неразрешимых проблем получать генерически неразрешимые проблемы, усиливая таким образом их неразрешимость.

Пусть I, J – некоторые множества с определенными на них функциями размера. *Клонирование* множества I в множестве J – это функция $C : I \to P(J)$, из I во множество всех подмножеств $P(J)$ множества J такая, что $\forall x, y \in I$

- либо $C(x) \cap C(y) = \varnothing$,
- либо $C(x) \subseteq C(y)$ или $C(y) \subseteq C(x)$.

Для множества $S \subseteq I$ определим *клон* $C(S)$ как объединение всех клонов элементов из S:

$$C(S) = \bigcup_{x \in S} C(x).$$

Будем называть клонирование C из I в J *эффективным*, если существует всюду определенная вычислимая функция $E : I \times \omega \to J$ такая, что для любого $x \in I$

$$C(x) = \{E(x, 0), E(x, 1), \ldots, \}.$$

Таким образом, с помощью алгоритма E можно эффективно перечислять все элементы каждого клона. Клонирование $C : I \to P(J)$ называется *непренебрежимым* (соответственно, *не сильно пренебрежимым*), если для любого $x \in I$ множество $C(x)$ не является пренебрежимым (соответственно, сильно пренебрежимым) в J.

Если $\mathcal{D} = (L, I)$ – проблема распознавания в I, то проблема распознавания $C(\mathcal{D}) = (C(L), J)$ в J называется *клоном* $\mathcal{D}$ в J относительно C. Из определения клонирования следует, что клонированные проблемы обладают следующим фундаментальным свойством:

- $x \in L \Longrightarrow C(x) \subseteq C(L)$,
- $x \notin L \Longrightarrow C(x) \subseteq \overline{C(L)}$.

Рассмотрим примеры клонирований для различных множеств входов.

Пусть $I = J = \{0,1\}^*$. Определим функцию $E : I \times \omega \to I$ следующим образом

$$E(a_1 \dots a_{n-1}a_n, i) = a_1 0 \dots a_{n-1} 0 a_n 1 bin(i),$$

где $a_k \in \{0,1\}$ и $bin(i)$ есть двоичная запись натурального числа i. Покажем, что E реализует полиномиально вычислимое непренебрежимое клонирование из I в I. Действительно, в том случае, если $x = a_1 \dots a_{n-1}a_n$, то

$$C(x) = \{a_1 0 \dots a_{n-1} 0 a_n 1 bin(i) \mid i \in \omega\}.$$

Поэтому $C(x) \cap C(y) = \varnothing$ для $x \neq y$, откуда C есть клонирование из I в I. Очевидно, C эффективное клонирование, более того, E вычислимо за полиномиальное время. Заметим, что для $I_m = \{w \in I \mid |w| = m\}$ имеем при $m \geq 2|x| + 1$:

$$\rho_m(C(x) \cap I_m) = \frac{2^{m-2|x|}}{2^m} = \frac{1}{2^{2|x|}} > 0.$$

Поэтому

$$\rho(C(x)) = \frac{1}{2^{2|x|}} > 0$$

и $C(x)$ непренебрежимо для любого x.

Рассмотрим клонирование в двухбуквенный алфавит. Для произвольного конечного алфавита A построим эффективное непренебрежимое

клонирование C_A из A^* в $\{0,1\}^*$. Зафиксируем инъективное отображение $\alpha : A \to \{0,1\}^*$ такое, что для любого $a \in A$ его образ $\alpha(a)$ имеет длину $[\log_2 |A|] + 1$. Отображение α продолжается до гомоморфизма моноидов $\alpha^* : A^* \to \{0,1\}^*$. Теперь легко видеть, что суперпозиция

$$C_A = C \circ \alpha^* : A^* \to P(\{0,1\}^*),$$

где C – клонирование, построенное в предыдущем примере, является непренебрежимым эффективным клонированием.

Интересный пример клонирования множеств натуральных чисел построили Джокуш и Шупп в [55]. Для любого $k \in \omega$ определим следующее множество

$$R_k = \{m \in \omega : 2^k \mid m,\ 2^{k+1} \nmid m\}.$$

В двоичной записи числа из R_k имеют нули в k младших разрядах, единицу в k+1-м разряде и произвольные цифры в остальных. Легко видеть, что при $n > k + 1$

$$\rho_n(R_k) = \frac{2^{n-k-1}}{2^n} = \frac{1}{2^{k+1}}.$$

Откуда

$$\rho(R_k) = \lim_{n\to\infty} \rho_n(R_k) = \lim_{n\to\infty} \frac{1}{2^{k+1}} = \frac{1}{2^{k+1}} > 0.$$

Кроме того, $R_k \cap R_l = \varnothing$ для всех $k \neq l$. Поэтому функция $R : k \mapsto R_k$ является эффективным непренебрежимым клонированием. Теперь для любого множества $S \subseteq \omega$ определим

$$\mathcal{R}(S) = \bigcup_{k\in S} R_k.$$

Это клонирование играет важную роль в [55], в частности при изучении генерического аналога тьюринговой сводимости.

Следующая теорема устанавливает важнейшую связь между классической разрешимостью алгоритмических проблем распознавания и генерической разрешимостью их клонов.

Теорема 2.1. *Пусть I, J – множества с функциями размера и $C : I \to P(J)$ – эффективное клонирование. Тогда для любой проблемы распознавания $\mathcal{D} = (L, I)$ в I имеет место следующее:*

1. *Если C непренебрежимое клонирование и проблема $C(\mathcal{D})$ генерически разрешима в J, то $\mathcal{D}$ разрешима в I.*

2. *Если C не сильно пренебрежимое клонирование и проблема $C(\mathcal{D})$ сильно генерически разрешима в J, то $\mathcal{D}$ разрешима I.*

Доказательство. Пусть $\mathcal{D} = (L, I)$ – проблема распознавания в I и $C(\mathcal{D}) = (C(L), J)$ – ее C-клон в J. Пусть $\mathcal{A}$ – генерический алгоритм, решающий $C(L)$ на J такой, что множество

$$G(\mathcal{A}) = \{x \in J : \mathcal{A}(x) \neq ?\}$$

генерическое. Построим алгоритм $\mathcal{B}$, который будет решать проблему $\mathcal{D}$ для всех входов. На входе $x \in I$ алгоритм $\mathcal{B}$ работает следующим образом.

1. Установить $i = 0$.

2. Вычислить $\mathcal{A}$ на элементе $E(x, i)$.

3. Если $\mathcal{A}(E(x, i)) \neq ?$ остановиться и выдать ответ.

4. Если $\mathcal{A}(E(x, i)) = ?$ положить $i = i + 1$ и вернуться на шаг 2.

Так как $C(x)$ непренебрежимо, то $C(x)$ имеет непустое пересечение с множеством $G(\mathcal{A})$. Поэтому, последовательно запуская алгоритм $\mathcal{A}$ на

элементах $E(x,0), E(x,1), \ldots$, мы найдем эффективно зависящее от x, число i_x такое, что $x' = E(x, i_x) \in G(\mathcal{A})$. Очевидно, $x \in L$ тогда и только тогда, когда $x' \in C(L)$ и тогда и только тогда, когда $\mathcal{A}$ выдает ответ ДА для x'. Поэтому мы можем эффективно решать $x \in L$, и пункт (1) доказан. Доказательство пункта (2) аналогично. □

Доказанную теорему можно переформулировать в терминах генерической неразрешимости.

Теорема 2.2. *Пусть I, J – множества с функциями размера и $C : I \to P(J)$ – эффективное клонирование. Тогда для любой неразрешимой проблемы распознавания $\mathcal{D} = (L, I)$ в I имеет место следующее:*

1. *Если C – непренебрежимое клонирование, то проблема $C(\mathcal{D})$ супер неразрешима в J.*

2. *Если C – не сильно пренебрежимое клонирование, то проблема $C(\mathcal{D})$ сильно неразрешима в J.*

Сформулируем это утверждение для вышеприведенных примеров клонирования.

Следствие 2.1. *Верно следующее:*

- *Пусть $I = \{0,1\}^*$ и C – клонирование из первого вышеприведенного примера. Тогда для любой неразрешимой проблемы $\mathcal{D} = (L, I)$ ее клонированная проблема $C(\mathcal{D}) = (C(L), I)$ супер неразрешима.*

- *Пусть A – конечный алфавит, $I = A^*$, и C_A – клонирование из второго вышеприведенного примера. Тогда для любой неразрешимой проблемы $\mathcal{D} = (L, I)$ ее клонированная проблема $C(\mathcal{D}) = (C(L), I)$ супер неразрешима.*

- *Если множество $S \subseteq \omega$ неразрешимо, то множество $\mathcal{R}(S)$ супер неразрешимо.*

2.2 Проблема остановки для машин Тьюринга

Будем рассматривать одноленточные машины Тьюринга с рабочим алфавитом $A = \{a_1, \ldots, a_m\}$. Программа P_M машины Тьюринга M с множеством внутренних состояний $Q = \{q_0, q_1, \ldots, q_n\}$ над алфавитом A состоит из правил вида

$$(q_i, a) \to (q_j, b, S)$$

– по одному правилу для всех ненулевых (нефинальных) состояний q_i и символов алфавита $a \in A$. Всего получается $n|A|$ правил. Под размером машины будем понимать число нефинальных внутренних состояний n. Будем отождествлять машины и их программы. Обозначим множество всех машин Тьюринга через $\mathcal{M}$.

Лемма 2.1. *Число машин Тьюринга размера n равно*

$$|\mathcal{M}_n| = \big(2(n+1)|A|\big)^{n|A|}.$$

Доказательство. Программа машины Тьюринга M с n внутренними со-

стояниями над алфавитом A имеет вид

$$(q_1, a_1) \to (q_{j_1}, b_1, S_1),\ 0 \le j_1 \le n,\ b_1 \in A,\ S_1 \in \{R, L\},$$
$$(q_1, a_2) \to (q_{j_2}, b_2, S_2),\ 0 \le j_2 \le n,\ b_2 \in A,\ S_2 \in \{R, L\},$$
$$\dots$$
$$(q_1, a_m) \to (q_{j_m}, b_m, S_m),\ 0 \le j_m \le n,\ b_m \in A,\ S_m \in \{R, L\},$$
$$(q_2, a_1) \to (q_{j_{m+1}}, b_{m+1}, S_{m+1}),\ 0 \le j_{m+1} \le n,\ b_{m+1} \in A,\ S_{m+1} \in \{R, L\},$$
$$\dots$$
$$(q_n, a_m) \to (q_{j_{nm}}, t_{nm}, S_{nm})\ 0 \le j_{nm} \le n,\ b_{nm} \in A,\ S_{nm} \in \{R, L\}.$$

Для каждого из $|A|$ правил, соответствующих состоянию q_i, существует $n + 1$ вариант выбора состояния q_j для перехода (учитываем также финальное состояние). Также есть $|A|$ вариантов для символа записи b и 2 варианта для сдвига. Перемножая эти количества для всех n состояний (по $|A|$ раз каждое), получаем нужный результат. □

Для произвольной машины M обозначим через $c(M)$ множество машин, которые получаются из M добавлением любого количества новых состояний и приписыванием к программе M произвольных правил для новых состояний.

Лемма 2.2. *Для любой машины Тьюринга M любая машина $M' \in c(M)$ вычисляет ту же функцию, что и M.*

Доказательство. Действительно, на любом входе новые состояния M' будут недостижимы и работа будет происходить фактически по программе машины M. □

Лемма 2.3. *Для любой машины M множество $c(M)$ не сильно пренебрежимо.*

Доказательство. Пусть машина M имеет k внутренних состояний и ее программа имеет вид

$$
\begin{aligned}
&(q_1, a_1) \to (q_{j_1}, b_1, S_1),\\
&(q_1, a_2) \to (q_{j_2}, b_2, S_2),\\
&\dots\\
&(q_1, a_m) \to (q_{j_m}, b_m, S_m),\\
&(q_2, a_1) \to (q_{j_{m+1}}, b_{m+1}, S_{m+1}),\\
&\dots\\
&(q_k, a_m) \to (q_{j_{km}}, b_{km}, S_{km}).
\end{aligned}
$$

Рассмотрим множество машин $c(M)_n$ размера $n > k$ с $n - k$ новыми состояниями. Программа машины $M' \in c(M)_n$ имеет вид

$$
\begin{aligned}
&(q_1, a_1) \to (q_{j_1}, b_1, S_1),\ j_1, b_1, S_1 \text{ фиксированы},\\
&(q_1, a_2) \to (q_{j_2}, b_2, S_2),\ j_2, b_2, S_2 \text{ фиксированы},\\
&\dots\\
&(q_1, a_m) \to (q_{j_m}, b_m, S_m),\ j_m, b_m, S_m \text{ фиксированы},\\
&(q_2, a_1) \to (q_{j_{m+1}}, b_{m+1}, S_{m+1}),\ j_{m+1}, b_{m+1}, S_{m+1} \text{ фиксированы},\\
&\dots\\
&(q_k, a_m) \to (q_{j_{km}}, b_{km}, S_{km}),\ j_{km}, b_{km}, S_{km} \text{ фиксированы},\\
&(q_{k+1}, a_1) \to (q_{j_{km+1}}, b_{km+1}, S_{km+1}),\ 0 \le j_{km+1} \le n,\ b_{km+1} \in A,\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad S_{km+1} \in \{L, R\},\\
&\dots\\
&(q_n, a_m) \to (q_{j_{nm}}, b_{nm}, S_{nm}),\ 0 \le j_{nm} \le n,\ b_{nm} \in A,\ S_{nm} \in \{L, R\}
\end{aligned}
$$

Здесь правила для первых k состояний зафиксированы и совпадают с программой машины M, а в остальных правые части выбираются произвольным образом. Посчитаем число таких машин. Аналогично тому,

как это делалось в доказательстве леммы 2.1, получаем

$$|c(M)_n| = \big(2(n+1)|A|\big)^{(n-k)|A|}.$$

Поэтому

$$\rho_n(c(M)) = \frac{|c(M)_n|}{|\mathcal{M}_n|} = \frac{\big(2(n+1)|A|\big)^{(n-k)|A|}}{\big(2(n+1)|A|\big)^{n|A|}} = \frac{1}{\big(2(n+1)|A|\big)^{k|A|}}.$$

Отсюда видно, что последовательность $\rho_n(c(M))$ не может стремиться к 0 экспоненциально быстро. Поэтому множество $c(M)$ не является сильно пренебрежимым. □

Лемма 2.4. *Определим функцию $C : \mathcal{M} \to P(\mathcal{M})$ следующим образом. Для любой машины Тьюринга M положим*

$$C(M) = c(M).$$

Тогда C является эффективным не сильно пренебрежимым клонированием.

Доказательство. Пусть M_1, M_2 – машины Тьюринга. Пусть $C(M_1) \cap C(M_2) \neq \varnothing$ и $M \in C(M_1) \cap C(M_2)$. Это означает, что программы машин M_1 и M_2 являются начальными частями программы машины M. А это, значит, что либо программа M_1 является начальной частью программы M_2 и тогда $C(M_2) \subseteq C(M_1)$, либо программа M_2 является начальной частью программы M_1 и тогда $C(M_1) \subseteq C(M_2)$. Таким образом, функция C является клонированием.

Эффективность клонирования C следует из того, что существует алгоритм, который для любой машины M перечисляет программы машин из $c(M)$ в порядке возрастания размера, а программы с одинаковым размером – в лексикографическом порядке.

То, что клонирование C является не сильно пренебрежимым, следует из леммы 2.3. □

Машина Тьюринга называется *нормализованной*, если из любого нефинального состояния q_i в соответствующих ему правилах в программе возможен переход в состояния q_j, где $j \leq |A|i + 1$. Смысл этого ограничения соответствует процессу экономного написания программы: начиная с первого правила, в каждом очередном новом правиле можно либо переходить в одно из старых состояний, либо в одно новое. Следующая лемма говорит о том, что класс вычислимых функций при этом не сужается.

Лемма 2.5. *Для любой машины Тьюринга M существует нормализованная машина Тьюринга M', вычисляющая ту же функцию.*

Доказательство. Заметим, что если в машине переименовать нефинальные состояния q_i в q_j и наоборот, а также в программе поменять местами соответствующие им правила, то работа машины на любом входе не изменится. Поэтому начиная с первого состояния можно нормализовывать все правила, переименовывая если нужно состояния, в которые происходят переходы. □

Обозначим множество всех нормализованных машин Тьюринга через $\mathcal{NM}$.

Лемма 2.6. *Число нормализованных машин Тьюринга размера n равно*

$$|\mathcal{NM}_n| = (2|A|)^{n|A|}(n+1)^{|A|(n-[(n-1)/|A|])} \prod_{i=1}^{[(n-1)/|A|]} (|A|i+2)^{|A|}.$$

Здесь через $[x]$ обозначается целая часть числа x.

Доказательство. Программа нормализованной машины Тьюринга M с n внутренними состояниями над алфавитом A имеет вид

$(q_1, a_1) \to (q_{j_1}, b_1, S_1),\ 0 \le j_1 \le |A| + 1,\ b_1 \in A,\ S_1 \in \{R, L\},$

$\dots$

$(q_i, a_1) \to (q_{j_i}, b_i, S_i),\ 0 \le j_i \le |A|i + 1 \le n,\ b_i \in A,\ S_i \in \{R, L\},$

$\dots$

$(q_l, a_1) \to (q_{j_l}, b_l, S_l),\ 0 \le j_l \le n,\ |A|l + 1 \ge n,\ b_l \in A,\ S_l \in \{R, L\},$

$\dots$

$(q_n, a_m) \to (q_{j_{nm}}, b_{nm}, S_{nm})\ 0 \le j_{nm} \le n,\ b_{nm} \in A,\ S_{nm} \in \{R, L\}.$

Для каждого из $|A|$ правил, соответствующих состоянию q_i, если $|A|i + 1 \le n$, существует $|A|i+2$ варианта выбора состояния для перехода (учитываем также финальное состояние). Для состояния q_i, где $|A|i + 1 \ge n$, существует $n+1$ вариант выбора состояния для перехода. Также есть $|A|$ вариантов для символа записи и 2 варианта для сдвига. Перемножая эти количества для всех n состояний (по $|A|$ раз каждое), получаем нужный результат. □

Пусть M – нормализованная машина Тьюринга, обозначим через $cn(M)$ множество машин, которые получаются из M добавлением любого количества новых состояний и приписыванием к программе M произвольных правил для новых состояний так, чтобы машина оставалась нормализованной.

Лемма 2.7. *Для любой нормализованной машины Тьюринга M любая машина $M' \in cn(M)$ вычисляет ту же функцию, что и M.*

Доказательство. Аналогично доказательству леммы 2.2. □

Лемма 2.8. *Для любой нормализованной машины M множество $c(M)$ не пренебрежимо.*

Доказательство. Пусть нормализованная машина M имеет k внутренних состояний и ее программа имеет вид

$(q_1, a_1) \to (q_{j_1}, b_1, S_1),\ 0 \le j_1 \le |A| + 1,\ b_1 \in A,\ S_1 \in \{R, L\},$

$\dots$

$(q_i, a_1) \to (q_{j_i}, b_i, S_i),\ 0 \le j_i \le |A|i + 1 \le k,\ b_i \in A,\ S_i \in \{R, L\},$

$\dots$

$(q_l, a_1) \to (q_{j_l}, b_l, S_l),\ 0 \le j_l \le n,\ |A|l + 1 \ge k,\ b_l \in A,\ S_l \in \{R, L\},$

$\dots$

$(q_k, a_m) \to (q_{j_{km}}, b_{km}, S_{km})\ 0 \le j_{km} \le k,\ b_{km} \in A,\ S_{km} \in \{R, L\}.$

Рассмотрим множество нормализованных машин $cn(M)_n$ размера $n > (k+2)|A|$ с $n-k$ новыми состояниями. Программа машины $M' \in cn(M)_n$ имеет вид

$(q_1, a_1) \to (q_{j_1}, b_1, S_1),\ j_1, b_1, S_1$ фиксированы,

$\dots$

$(q_1, a_m) \to (q_{j_m}, b_m, S_m),\ j_m, b_m, S_m$ фиксированы,

$(q_2, a_1) \to (q_{j_{m+1}}, b_{m+1}, S_{m+1}),\ j_{m+1}, b_{m+1}, S_{m+1}$ фиксированы,

$\dots$

$(q_k, a_m) \to (q_{j_{km}}, b_{km}, S_{km}),\ j_{km}, b_{km}, S_{km}$ фиксированы,

$(q_{k+1}, a_1) \to (q_{j_{km+1}}, b_{km+1}, S_{km+1}),\ 0 \le j_{km+1} \le |A|(k+1) + 1 \le n,$
$b_{km+1} \in A,\ S_{km+1} \in \{L, R\},$

$\dots$

$(q_i, a_1) \to (q_{j_i}, b_i, S_i),\ 0 \le j_i \le |A|i + 1 \le n,\ b_i \in A,\ S_i \in \{R, L\},$

$\dots$

$(q_l, a_1) \to (q_{j_l}, b_l, S_l),\ 0 \le j_l \le n,\ |A|l + 1 \ge n,\ b_l \in A,\ S_l \in \{R, L\},$

$\dots$

$(q_n, a_m) \to (q_{j_{nm}}, b_{nm}, S_{nm}),\ 0 \le j_{nm} \le n,\ b_{nm} \in A,\ S_{nm} \in \{L, R\}$

Здесь правила для первых k состояний зафиксированы и совпадают с программой машины M, а в остальных правые части выбираются произвольным образом. Посчитаем число таких машин. Оно равно числу фрагментов программ для новых состояний. Аналогично тому, как это делалось в доказательстве леммы 2.6, получаем

$$|cn(M)_n| = (2|A|)^{(n-k)|A|}(n+1)^{|A|(n-[(n-1)/|A|])}\prod_{i=k+1}^{[(n-1)/|A|]}(|A|i+2)^{|A|}.$$

Поэтому

$$\rho(cn(M)) = \lim_{n\to\infty}\frac{|cn(M)_n|}{|\mathcal{NM}_n|} =$$

$$= \lim_{n\to\infty}\frac{(2|A|)^{(n-k)|A|}(n+1)^{|A|(n-[(n-1)/|A|])}\prod_{i=k+1}^{[(n-1)/|A|]}(|A|i+2)^{|A|}}{(2|A|)^{n|A|}(n+1)^{|A|(n-[(n-1)/|A|])}\prod_{i=1}^{[(n-1)/|A|]}(|A|i+2)^{|A|}} =$$

$$= \lim_{n\to\infty}\frac{1}{(2|A|)^{k|A|}\prod_{i=1}^{k}(|A|i+2)^{|A|}} = \lim_{n\to\infty} const = const > 0.$$

□

Лемма 2.9. *Определим функцию $CN : \mathcal{NM} \to P(\mathcal{NM})$ следующим образом. Для любой нормализованной машины Тьюринга M положим*

$$CN(M) = cn(M).$$

Тогда CN является эффективным непренебрежимым клонированием.

Доказательство. Доказательство аналогично доказательству леммы 2.4. То, что клонирование CN является непренебрежимым, следует из леммы 2.8. □

Рассмотрим теперь один из вариантов классической проблемы остановки. Зафиксируем алфавит $A = \{0, 1, \square\}$, где $\square$ – пустой символ. Определим следующее множество

$$HP_{\mathcal{M}} = \{M \in \mathcal{M} : M \text{ останавливается на входе } 0\}.$$

Аналогично определим

$$HP_{\mathcal{NM}} = \{M \in \mathcal{NM} : M \text{ останавливается на входе } 0\}.$$

Проблемы $(HP_{\mathcal{M}}, \mathcal{M})$ и $(HP_{\mathcal{NM}}, \mathcal{NM})$ в классическом смысле являются алгоритмически неразрешимыми. Следующая теорема утверждает, что они остаются неразрешимыми и в генерическом смысле.

Теорема 2.3. *Верно следующее:*

1. *Проблема* $(HP_{\mathcal{M}}, \mathcal{M})$ *сильно неразрешима.*

2. *Проблема* $(HP_{\mathcal{NM}}, \mathcal{NM})$ *супер неразрешима.*

Доказательство. Докажем пункт 1. Заметим, что $C(HP_{\mathcal{M}}) = HP_{\mathcal{M}}$, так как по лемме 2.2, если $M \in HP_{\mathcal{M}}$, то для любой машины $M' \in C(M)$ имеет место $M' \in HP_{\mathcal{M}}$, а значит $C(M) \subseteq HP_{\mathcal{M}}$. То есть, проблема остановки является клоном самой себя:

$$(C(HP_{\mathcal{M}}), \mathcal{M}) = (HP_{\mathcal{M}}, \mathcal{M}).$$

А так как, по лемме 2.4, клонирование C является эффективным и не сильно пренебрежимым, то проблема $(HP_{\mathcal{M}}, \mathcal{M})$ сильно неразрешима по теореме 2.2.

Доказательство пункта 2 аналогично с использованием клонирования CN и леммы 2.9. □

Аналогичным образом можно доказывать генерическую неразрешимость других вариантов проблемы остановки и других неразрешимых проблем, связанных с программами машин Тьюринга.

В качестве еще одного приложения развитой техники, докажем генерический аналог теоремы Клини о неподвижной точке. Классическая

теорема Клини о неподвижной точке [66], известная также как теорема Клини о рекурсии, утверждает, что для любого алгоритмически вычислимого отображения множества программ машин Тьюринга на множество программ машин Тьюринга существует неподвижная точка: найдется такая машина, что и она и ее образ под действием этого отображения вычисляют одну и ту же функцию. Докажем генерический аналог этой теоремы: для любого алгоритмически вычислимого отображения генерического рекурсивного подмножества программ машин Тьюринга на множество программ машин Тьюринга существует неподвижная точка.

Теорема 2.4. *Имеет место следующее:*

1. *Для любого генерического алгоритма* $\mathcal{A} : \mathcal{NM} \to \mathcal{NM} \cup \{?\}$ *существует такая машина Тьюринга* M*, что* $\mathcal{A}(M) \neq ?$ *и машина* $\mathcal{A}(M)$ *вычисляет ту же функцию, что и* M*.*

2. *Для любого сильно генерического алгоритма* $\mathcal{A} : \mathcal{M} \to \mathcal{M} \cup \{?\}$ *существует такая машина Тьюринга* M*, что* $\mathcal{A}(M) \neq ?$ *и машина* $\mathcal{A}(M)$ *вычисляет ту же функцию, что и* M*.*

Доказательство. Докажем пункт 1, пункт 2 доказывается аналогично. Допустим противное, то есть, что существует такой генерический алгоритм $\mathcal{A} : \mathcal{NM} \to \mathcal{NM} \cup \{?\}$, что для любой машины Тьюринга M если $\mathcal{A}(M) \neq ?$, то $\mathcal{A}(M)$ вычисляет функцию, отличную от той, что вычисляет M. Построим теперь обычный алгоритм $\mathcal{B} : \mathcal{NM} \to \mathcal{NM}$, определенный для любой машины Тьюринга, но не имеющий неподвижной точки. Это будет противоречить классической теореме о неподвижной точке. Алгоритм $\mathcal{B}$ будет работать на машине M следующим образом. Перебирает множество $c(M)$ в порядке возрастания размеров машин и

на каждой машине запускает алгоритм $\mathcal{A}$ до тех пор пока не найдет машину M^* такую, что $\mathcal{A}(M^*) \neq ?$. Это всегда произойдет, так как, по лемме 2.8, множество $c(M)$ не является пренебрежимым, а множество $\{M \in \mathcal{NM} : \mathcal{A}(M) = ?\}$ пренебрежимо. По предположению, машина $\mathcal{A}(M^*)$ вычисляет функцию, отличную от функции, вычисляемой машиной M^*, которая вычисляет ту же функцию, что и M. Поэтому алгоритм $\mathcal{B}$ не имеет неподвижной точки. Противоречие. □

2.3 Полугруппы с генерически неразрешимой проблемой равенства слов

В этом подразделе мы покажем, как можно амплифицировать генерическую сложность конечно определенных полугрупп с неразрешимой проблемой равенства слов. Именно, мы опишем общий метод построения конечно определенных полугрупп с супер неразрешимой проблемой равенства слов. Более точно, по конечно определенной полугруппе $\mathfrak{S}$ мы строим полугруппу $\mathfrak{S}_x$, чья проблема равенства слов является эффективным непренебрежимым колонированием проблемы равенства слов полугруппы $\mathfrak{S}$.

Пусть

$$\mathfrak{S} = \langle a_1, \ldots, a_n | r_1 = s_1, \ldots, r_k = s_k \rangle$$

— конечно определенная полугруппа с множеством порождающих

$$A = \{a_1, \ldots, a_n\}$$

и множеством определяющих соотношений

$$R = \{r_1 = s_1, \ldots, r_k = s_k\}.$$

Обозначим через $WP_{\mathfrak{S}}$ проблему равенства слов в полугруппе $\mathfrak{S}$, то

есть

$$WP_{\mathfrak{S}} = \{(u, v) \in A^* \times A^* \mid u = v \text{ в } \mathfrak{S}\}.$$

Для символа $x \notin A$ положим

$$\mathfrak{S}_x = \langle A, x | R, x = xa_1, \ldots, x = xa_n, x = xx \rangle.$$

Таким образом, $\mathfrak{S}_x$ также является конечно определенной полугруппой. Обозначим $A_x = A \cup \{x\}$.

Лемма 2.10. *Для любых слов $w_1, w_2 \in A^*$ и $v_1, v_2 \in A_x^*$ имеет место:*

1. $w_1 = w_2$ *в* $\mathfrak{S} \Leftrightarrow w_1 = w_2$ *в* $\mathfrak{S}_x$,

2. $w_1 = w_2$ *в* $\mathfrak{S} \Leftrightarrow w_1 x v_1 = w_2 x v_2$ *в* $\mathfrak{S}_x$.

Доказательство. В полугруппе $\mathfrak{S}_x$ для слов, которые не содержат x, применимы только определяющие соотношения из R. Это доказывает пункт 1.

С другой стороны, если слово $w \in A_x^*$ содержит символ x, то, используя соотношения $x = xa_i$ и $x = xx$ из $\mathfrak{S}_x$, можно удалить все символы в w справа от x в w. Это показывает, что для любых $w_1, w_2 \in A^*$ и $v_1, v_2 \in A_x^*$ имеет место

$$w_1 x v_1 = w_2 x v_2 \text{ в } \mathfrak{S}_x \Leftrightarrow w_1 x = w_2 x \text{ в } \mathfrak{S}_x.$$

Теперь, к словам $w_1 x$, $w_2 x$ применимы только соотношения из R. Таким образом,

$$w_1 x = w_2 x \text{ в } \mathfrak{S}_x \Leftrightarrow w_1 = w_2 \text{ в } \mathfrak{S}.$$

Пункт 2 доказан. □

Следствие 2.2. *Каноническое вложение $A \to A_x$ продолжается до вложения полугрупп $\mathfrak{S} \to \mathfrak{S}_x$.*

Лемма 2.11. *Пусть* $I = A^* \times A^*$ *и* $J = A_x^* \times A_x^*$. *Для* $(u, v) \in I$ *положим*

$$C(u, v) = \{(uxp, vxq) \mid p, q \in A_x^*\}.$$

Тогда C *является эффективным непренебрежимым клонированием.*

Доказательство. Для того, чтобы доказать, что отображение C является клонированием, достаточно заметить, что

$$C(u, v) \cap C(u_1, v_1) \neq \varnothing \Longleftrightarrow u = u_1 \text{ and } v = v_1,$$

что очевидно.

Докажем, что C является эффективным клонированием. Действительно, определим

$$\tau : \omega \to A_x^* \times A_x^*,$$

как вычислимое биективное перечисление всех пар слов алфавита A_x. Заметим, что для любого $i \in \omega$

$$\tau(i) = (\tau_1(i), \tau_2(i))$$

для некоторых фиксированных вычислимых функций

$$\tau_1 : \omega \to A_x^*, \tau_2 : \omega \to A_x^*.$$

Теперь функция $E : A^* \times A^* \times \omega \to A_x^* \times A_x^*$ определенная как

$$E(u, v, i) = (ux\tau_1(i), vx\tau_2(i))$$

дает эффективное перечисление клонов $C(u, v)$, $(u, v) \in I$.

Докажем теперь, что клонирование C непренебрежимо. По определению, размер пары $(p, q) \in J$ равен $|p| + |q|$, поэтому сфера J_n радиуса n в J есть следующее множество

$$J_n = \{(p, q) \in A_x^* \times A_x^* \mid |p| + |q| = n\}.$$

Поэтому

$$|J_n| = \sum_{i=0}^{n} |A_x|^i |A_x|^{n-i} = (n+1)|A_x|^n.$$

Аналогично, для $n \geq |u| + |v| + 2$

$$|C(u,v) \cap J_n| = \sum_{i=0}^{n-|u|-|v|-2} |A_x|^i |A_x|^{n-i} =$$

$$= (n - |u| - |v| - 1)|A_x|^{n-|u|-|v|-2}.$$

Отсюда

$$\rho_n(C(u,v) \cap J_n) = \frac{(n - |u| - |v| - 1)|A_x|^{n-|u|-|v|-2}}{(n+1)|A_x|^n}.$$

Переходя к пределу, получаем

$$\rho(C(u,v)) = \lim_{n\to\infty} \frac{(n - |u| - |v| - 1)|A_x|^{n-|u|-|v|-2}}{(n+1)|A_x|^n} = \frac{1}{|A_x|^{|u|+|v|+2}} > 0.$$

Это доказывает, что C непренебрежимо. □

Лемма 2.12. *Для любой конечно определенной полугруппы $\mathfrak{S}$ проблема равенства слов в $\mathfrak{S}_x$ является C-клоном проблемы равенства слов в $\mathfrak{S}$:*

$$WP_{\mathfrak{S}_x} = C(WP_{\mathfrak{S}}).$$

Доказательство. Это следует непосредственно из леммы 2.10. □

Теорема 2.5. *Если проблема равенства слов в полугруппе $\mathfrak{S}$ неразрешима, то проблема равенства слов в полугруппе $\mathfrak{S}_x$ супер неразрешима.*

Доказательство. По лемме 2.11 отображение C является эффективным непренебрежимым клонированием. По лемме 2.12 имеем $WP_{\mathfrak{S}_x} = C(WP_{\mathfrak{S}})$. Теперь по теореме 2.2 если проблема равенства слов $WP_{\mathfrak{S}}$ неразрешима, то проблема равенства слов $WP_{\mathfrak{S}_x} = C(WP_{\mathfrak{S}})$ супер неразрешима. □

В 1956 г. Цейтин построил полугруппу $\mathfrak{T}$, заданную 5 порождающими и 7 определяющими соотношениями с неразрешимой проблемой равенства слов [25]:

$$\mathfrak{T} = \langle a, b, c, d, e \mid ca, ad = da, bc = cb,$$

$$bd = db, ce = eca, de = edb, cca = ccae \rangle.$$

В данном случае супер неразрешимая полугруппа $\mathfrak{T}_x$ будет иметь 6 порождающих и 13 определяющих соотношений, чья общая длина равна 49.

Остается открытым вопрос о существовании конечно определенной группы с супер неразрешимой проблемой равенства слов. По-видимому, метод генерической амплификации для построения такой группы из известных конечно определенных групп с неразрешимой проблемой равенства слов, неприменим, так как в [82] было доказано, что в любой конечно определенной группе множество слов, равных единице, является пренебрежимым.

2.4 Десятая проблема Гильберта

В данном подразделе будет изучена генерическая разрешимость десятой проблемы Гильберта для двух способов представления диофантовых уравнений.

Диофантовым уравнением называется уравнение вида

$$P(x_1, x_2, \ldots, x_n) = 0,$$

где P – многочлен с целыми коэффициентами. По техническим соображениям мы будем рассматривать диофантовы уравнения с фиксированным числом неизвестных $m \geq 13$. Любое диофантово уравнение

$P(x_1, \ldots, x_m) = 0$ где P – полином с целыми коэффициентами фактически задается полиномом P. Такие полиномы можно представлять с помощью так называемых арифметических схем. *Арифметическая схема* от переменных $x_1, \ldots, x_m$ состоит из конечного числа промежуточных переменных $x_{m+1}, \ldots, x_{n+m}$ и присваиваний (одно присваивание для каждой промежуточной переменной) одного из следующих типов:

- $x_i = x_j * x_k$, где $j, k < i$ и $* \in \{+, -, \times\}$,
- $x_i = x_j + 1$, где $j < i$,
- $x_i = x_j - 1$, где $j < i$.

Переменные $x_1, \ldots, x_m$ называются *входными переменными* схемы. Последняя промежуточная переменная x_{m+n} называется *выходной переменной* схемы. *Размер* схемы – это число присваиваний n.

Легко видеть, что каждая промежуточная переменная схемы получается с помощью сложения, вычитания, умножения и увеличения (уменьшения) на 1 из предыдущих переменных. Поэтому выходная переменная схемы является целочисленным полиномом от входных переменных. И наоборот, любой целочисленный полином от переменных $x_1, \ldots, x_m$ можно представить с помощью некоторой схемы с входными $x_1, \ldots, x_m$. Например, полином $x_1^3 + 2x_1x_2 - 1$ можно представить так:

$$x_3 = x_1x_1,\ x_4 = x_3x_1,\ x_5 = x_1x_2,$$

$$x_6 = x_5 + x_5,\ x_7 = x_4 + x_6,\ x_8 = x_7 - 1.$$

Таким образом, мы будем отождествлять диофантово уравнение $P(x_1, \ldots, x_m) = 0$ с арифметической схемой, представляющей полином P. Обозначим через $\mathcal{D}$ множество всех диофантовых уравнений, пред-

ставленных с помощью арифметических схем. Имея такое представление, легко подсчитать число диофантовых уравнений фиксированного размера.

Лемма 2.13. *Число всех диофантовых уравнений размера n с m неизвестными есть*

$$|\mathcal{D}_n| = \prod_{i=m}^{n+m} (3(i-1)^2 + 2(i-1)).$$

Доказательство. Присваивание для переменной x_i может иметь вид $x_i = x_j * x_k$, где $j, k < i$ и $* \in \{+, -, \times\}$. Мы имеем $3(i-1)^2$ варианта выбора для такого присваивания. Оно может быть также вида $x_i = x_j + 1$ или $x_i = x_j - 1$, где $j < i$. Имеется $2(i-1)$ варианта для этого случая. Итого есть $3(i-1)^2 + 2(i-1)$ варианта для выбора i-го присваивания. А для всех n присваиваний схемы мы имеем

$$\prod_{i=m}^{n+m} (3(i-1)^2 + 2(i-1))$$

вариантов. □

Полином P можно представлять многими арифметическими схемами. Пусть имеется схема Σ размера k, вычисляющая полином P. Предположим, что последнее присваивание Σ имеет вид $x_{k+m} = x_i * x_j$ (для других типов присваивания доказательство аналогично) Для любого $n > k$ можно построить много схем, вычисляющих P следующим образом. Первые k присваиваний оставляем такими же, как и в Σ, последнее будет $x_{n+m} = x_i * x_j$ (правая часть как в Σ), а остальные $n-k-1$ присваиваний произвольные. Легко видеть, что такие схемы вычисляют P потому что их выходная переменная такая же, как в Σ и не зависит от новых добавленных присваиваний. Обозначим через $as(P)$ множество всех таких схем.

Лемма 2.14. *Для любого полинома P и $n > m$*

$$\frac{|as(P)_n|}{|\mathcal{D}_n|} > \frac{1}{Cn^2}$$

для некоторой константы $C > 0$, зависящей только от P. Таким образом, множество $as(P)$ не является сильно пренебрежимым.

Доказательство. Посчитаем число схем из $as(P)$. Первые k присваиваний фиксированы, последнее тоже, но остальные $n-k-1$ присваиваний могут быть произвольными. Как и в доказательстве леммы 2.13 мы можем посчитать

$$|as(P)_n| = \prod_{i=k+1}^{n+m-1} (3(i-1)^2 + 2(i-1)).$$

Теперь можно оценить:

$$\frac{|as(P)_n|}{|\mathcal{D}_n|} = \frac{\prod_{i=k+1}^{n+m-1}(3(i-1)^2 + 2(i-1))}{\prod_{i=m}^{n+m}(3(i-1)^2 + 2(i-1))} =$$

$$= \frac{1}{(3(m+n-1)^2 + 2(m+n-1)) \prod_{i=m}^{k}(3(i-1)^2 + 2(i-1))} =$$

$$= \frac{1}{C(3(m+n-1)^2 + 2(m+n-1))} > \frac{1}{6Cn^2},$$

где $C = \prod_{i=m}^{k}(3(i-1)^2 + 2(i-1))$ – константа. □

Определим функцию $CD : \mathcal{D} \to P(\mathcal{D})$ следующим образом. Для любого диофантова уравнения $P \in \mathcal{D}$ положим

$$CD(P) = as(P).$$

Лемма 2.15. *Функция $CD : \mathcal{D} \to P(\mathcal{D})$ является эффективным не сильно пренебрежимым клонированием.*

Доказательство. Пусть $P_1, P_2 \in \mathcal{D}$ – арифметические схемы, представляющие диофантовы уравнения. Пусть $as(P_1) \cap as(P_2) \neq \varnothing$ и $P \in$

$as(P_1) \cap as(P_2)$. Тогда схема P является продолжением и схемы P_1 и схемы P_2. Но это означает, что либо P_1 продолжает схему P_2 и тогда $P_1 \in as(P_2)$, откуда $as(P_1) \subseteq as(P_2)$, либо P_2 продолжает схему P_1 и тогда $as(P_2) \subseteq as(P_1)$. Таким образом, функция CD является клонированием.

То, что CD является эффективным клонированием, следует из существования алгоритма, который по любой схеме P перечисляет множество $as(P)$ в порядке возрастания размеров, а схемы одинакового размера в лексикографическом порядке.

То, что клонирование CD не сильно пренебрежимо, следует из леммы 2.14. □

Теперь определим множество

$$SOL(\mathcal{D}) = \{P \in \mathcal{D} : \exists x_1 \in \mathbb{Z} \ldots \exists x_m \in \mathbb{Z}\ P(x_1, \ldots, x_m) = 0\}.$$

Из неразрешимости десятой проблемы Гильберта следует, что проблема распознавания $(SOL(\mathcal{D}), \mathcal{D})$ неразрешима в классическом смысле. Следующая теорема утверждает, что она неразрешима и в генерическом смысле.

Теорема 2.6. *Проблема $(SOL(\mathcal{D}), \mathcal{D})$ сильно неразрешима.*

Доказательство. Заметим, что, если $P \in SOL(\mathcal{D})$, то $as(P) \subseteq SOL(\mathcal{D})$. Поэтому для клонирования CD имеет место

$$(CD(SOL(\mathcal{D})), \mathcal{D}) = (SOL(\mathcal{D}), \mathcal{D}).$$

А так, как по лемме 2.15, клонирование CD эффективно и не сильно пренебрежимо, то, по теореме 2.2 проблема $(SOL(\mathcal{D}), \mathcal{D})$ сильно неразрешима. □

Следующая теорема показывает, что «плохие» входы, то есть входы, на которых любой генерический алгоритм, решающий десятую проблему Гильберта, выдает ответ «?», можно эффективно генерировать.

Теорема 2.7. *Для любого генерического полиномиального алгоритма $\mathcal{A}$, решающего проблему $(SOL(\mathcal{D}), \mathcal{D})$, существует полиномиальный вероятностный алгоритм $\mathcal{B}$, генерирующий вход размера n из множества*

$$BH(\mathcal{A}) = \{P \in \mathcal{D} : \mathcal{A}(P) =?\}$$

с вероятностью $> 1 - \frac{1}{e^n}$.

Доказательство. Допустим, что существует генерический полиномиальный алгоритм $\mathcal{A}$, решающий десятую проблему Гильберта на полиномиальном генерическом множестве

$$G = \{P \in \mathcal{D} : \mathcal{A}(P) \neq ?\}.$$

Тогда для достаточно большого n

$$\frac{|BH(\mathcal{A})_n|}{|A_n|} = \frac{|A_n \setminus G_n|}{|A_n|} > \frac{1}{n^3}. \tag{1}$$

Действительно, допустим противное, т.е. что для бесконечно многих n

$$\frac{|A_n \setminus G_n|}{|A_n|} < \frac{1}{n^3}. \tag{2}$$

Тогда мы можем алгоритмически решить десятую проблему Гильберта для любого диофантового уравнения $P = 0$, представленного схемой $\mathcal{C}$, следующим образом:

1. Если $\mathcal{C} \in G$, то решаем алгоритмом $\mathcal{A}$.
2. Если $\mathcal{C} \notin G$, то генерируем схемы из $as(P)$ до тех пор, пока не получим схему $\mathcal{C}^* \in G$. После этого решаем $\mathcal{C}^*$ алгоритмом $\mathcal{A}$.

Этот алгоритм решает десятую проблему Гильберта и всегда останавливается, так как оценка из леммы 2.14

$$\frac{|as(P)_n|}{|A_n|} > \frac{1}{Cn^2}$$

и оценка (2) влекут, что $as(P)_k \cap G_k \neq \varnothing$ для достаточно большого k. Это противоречие доказывает (1).

Теперь полиномиальный вероятностный алгоритм $\mathcal{B}$, генерирующий входы из $BH(\mathcal{A})$, работает на n следующим образом.

1. Повторить n^4 раз:
   (a) Сгенерировать случайно и равномерно вход x размера n.
   (b) Если $x \notin G$, то остановиться и выдать x.
2. Выдать ответ «?».

Оценка (1) влечет, что вероятность шага 2 в алгоритме $\mathcal{B}$ меньше, чем

$$\left(1 - \frac{1}{n^3}\right)^{n^4} = \left(\left(1 - \frac{1}{n^3}\right)^{n^3}\right)^n \sim e^{-n}.$$

Таким образом, вероятность того, что алгоритм $\mathcal{B}$ выдаст вход из $BH(\mathcal{A})$ больше, чем $1 - \frac{1}{e^n}$. □

Система диофантовых уравнений записана в *форме Сколема* [79, 13], если каждое уравнение в ней имеет один из следующих типов:

1. $x_i = x_j x_k$,
2. $x_i = x_j + x_k$,
3. $x_i = 1$.

Нетрудно показать, что для любого диофантова уравнения (системы) можно эффективно построить эквивалентную ему систему диофантовых

уравнений в форме Сколема. Например, для уравнения $x^2 - 3y^2 = 1$ эквивалентной системой Сколема является

$$x_1 = x_2 x_2,\ x_3 = x_4 x_4,\ x_5 = x_3 + x_3,$$

$$x_6 = x_5 + x_3,\ x_7 = 1,\ x_8 = x_6 + x_7,\ x_8 = x_1.$$

Будем называть систему в форме Сколема *нормализованной*, если в k-м уравнении системы могут встречаться только переменные x_i, где $i \leq 3k$. Очевидно, что любую систему в форме Сколема можно нормализовать при помощи подходящего перенумерования переменных.

Рассмотрим теперь проблему разрешимости нормализованных систем диофантовых уравнений в форме Сколема. Размер такой системы — это число уравнений в ней. Обозначим через $\mathcal{S}$ множество всех нормализованных систем диофантовых уравнений в форме Сколема.

Лемма 2.16. *Число нормализованных систем в форме Сколема размера n есть*

$$|\mathcal{S}_n| = \prod_{k=1}^{n} (54k^3 + 3k).$$

Доказательство. Для k-го уравнения в системе $S \in \mathcal{S}_n$ существует $(3k)^3$ вариантов выбрать уравнение вида $x_i = x_j x_k$, также $(3k)^3$ вариантов выбрать уравнение вида $x_i = x_j + x_k$ и только $3k$ вариантов выбрать уравнение вида $x_i = 1$. Итого для k-го уравнения есть $54k^3 + 3k$ вариантов. А для всей системы из n уравнений имеем

$$|\mathcal{S}_n| = \prod_{k=1}^{n} (54k^3 + 3k)$$

вариантов. □

Теперь определим множество

$$SOL_{\mathbb{N}}(\mathcal{S}) = \{S \in \mathcal{S} : \text{система } S \text{ разрешима над } \mathbb{N}\}.$$

Напомним, что множество натуральных чисел $\mathbb{N}$ не содержит 0. Из неразрешимости десятой проблемы Гильберта следует, что проблема распознавания $(SOL_{\mathbb{N}}(\mathcal{S}), \mathcal{D})$ неразрешима в классическом смысле. Приведем полиномиальный генерический алгоритм для проверки разрешимости диофантовых уравнений в форме Сколема над множеством натуральных чисел $\mathbb{N}$.

Теорема 2.8. *Проблема разрешимости диофантовых уравнений в форме Сколема $(SOL_{\mathbb{N}}(\mathcal{S}), \mathcal{D})$ генерически разрешима за полиномиальное время.*

Доказательство. Соответствующий генерический полиномиальный алгоритм работает на системе S следующим образом:

1. ищет в системе S уравнение вида $x_i = x_i + x_j$;
2. если такое уравнение найдется, то система неразрешима над $\mathbb{N}$ и алгоритм выдает ответ 0;
3. иначе выдает ответ «?».

Покажем, что этот алгоритм дает неопределенный ответ на пренебрежимом множестве систем Сколема. Обозначим через $\mathcal{L}$ множество систем, в которых отсутствуют уравнения вида $x_i = x_i + x_j$. Число вариантов выбрать k-е уравнение в системе из множества $\mathcal{L}$ есть $54k^3 + 3k - 9k^2$. Поэтому

$$|\mathcal{L}_n| = \prod_{k=1}^{n} (54k^3 + 3k - 9k^2).$$

По лемме 2.16

$$|\mathcal{S}_n| = \prod_{k=1}^{n} (54k^3 + 3k),$$

поэтому

$$\rho_n(\mathcal{L}) = \frac{|\mathcal{L}_n|}{|\mathcal{S}_n|} = \frac{\prod\limits_{k=1}^{n}(54k^3+3k-9k^2)}{\prod\limits_{k=1}^{n}(54k^3+3k)} =$$

$$= \prod_{k=1}^{n}\frac{18k^3+k-3k^2}{18k^3+k} = \prod_{k=1}^{n}\Big(1-\frac{3k^2}{18k^3+k}\Big) <$$

$$< \prod_{k=1}^{n}\Big(1-\frac{3k^2}{18k^3+k^3}\Big) = \prod_{k=1}^{n}\Big(1-\frac{3}{19k}\Big) < \prod_{k=1}^{n}\Big(1-\frac{1}{7k}\Big).$$

Для того чтобы оценить сверху последнее произведение, возведем его в степень 7:

$$\prod_{k=1}^{n}\Big(1-\frac{1}{7k}\Big)^7 < \prod_{k=1}^{n}\Big((1-\frac{1}{7k})(1-\frac{1}{7k+1})\dots(1-\frac{1}{7k+6})\Big) =$$

$$= \prod_{k=7}^{n}\Big(1-\frac{1}{k}\Big) = \prod_{k=7}^{n}\frac{k-1}{k} = \frac{6}{7}\cdot\frac{7}{8}\cdot\ldots\cdot\frac{n-2}{n-1}\cdot\frac{n-1}{n} = \frac{6}{n}.$$

Итого получаем

$$\rho(\mathcal{L}) = \lim_{n\to\infty}\frac{|\mathcal{L}_n|}{|\mathcal{S}_n|} \le \lim_{n\to\infty}\sqrt[7]{\frac{6}{n}} = 0.$$

Теорема доказана. □

Докажем, что проблема разрешимости нормализованных систем диофантовых уравнений в форме Сколема над множеством целых чисел является сильно неразрешимой.

Для произвольной нормализованной системы диофантовых уравнений в форме Сколема $S = \{S_1, \dots, S_m\}$ рассмотрим множество систем $\mathrm{eq}(S)$, которые получаются добавлением к системе S любого количества произвольных уравнений вида $x_i = x_j x_k$ или $x_i = x_j + x_k$, где $i, j, k > 3m$, с сохранением условия нормализации. Очевидно, что любая система из $\mathrm{eq}(S)$ разрешима в целых числах тогда и только тогда, когда разрешима в целых числах система S.

Лемма 2.17. *Для любой системы S множество $eq(S)$ не является сильно пренебрежимым.*

Доказательство. Пусть $n > m$. Для t-го добавленного к S уравнения вида $x_i = x_j x_k$ или $x_i = x_j + x_k$, где $i, j, k > 3m$, имеется $2(3t)^3 = 54t^3$ вариантов. Поэтому

$$|\mathrm{eq}(S)_n| = \prod_{t=1}^{n-m} (54t^3).$$

Теперь по лемме 2.16

$$\rho_n(\mathrm{eq}(S)) = \frac{|\mathrm{eq}(S)_n|}{|\mathcal{S}_n|} = \frac{\prod\limits_{k=1}^{n-m} (54k^3)}{\prod\limits_{k=1}^{n} (54k^3 + k)} =$$

$$= \prod_{k=1}^{n-m} \frac{54k^2}{54k^2+1} \prod_{k=n-m+1}^{n} \frac{1}{54k^3+k}.$$

Оценим снизу первое произведение:

$$\prod_{k=1}^{n-m} \frac{54k^2}{54k^2+1} = \prod_{k=1}^{n-m} \Big(1 - \frac{1}{54k^2+1}\Big) >$$

$$> \prod_{k=2}^{n-m+1} \Big(1 - \frac{1}{k^2}\Big) = \prod_{k=2}^{n-m+1} \frac{(k-1)(k+1)}{k^2} =$$

$$= \frac{1 \cdot 3}{2^2} \cdot \frac{2 \cdot 4}{3^2} \cdot \ldots \cdot \frac{(n-m-1)(n-m+1)}{(n-m)^2} \cdot \frac{(n-m)(n-m+2)}{(n-m+1)^2} =$$

$$= \frac{n-m+2}{2(n-m+1)} > \frac{1}{2}.$$

Теперь оценим второе произведение:

$$\prod_{k=n-m+1}^{n} \frac{1}{54k^3+k} > \frac{1}{(54n^3+n)^m}.$$

Итого получаем

$$\rho_n(\mathrm{eq}(S)) = \frac{|\mathrm{eq}(S)_n|}{|\mathcal{S}_n|} > \frac{1}{2(54n^3+n)^m}.$$

Из этого неравенства следует, что последовательность $\rho_n(\mathrm{eq}(S))$ не может стремиться к 0 экспоненциально быстро. Поэтому множество $\mathrm{eq}(S)$ не является сильно пренебрежимым. □

Определим функцию $CS : \mathcal{S} \to P(\mathcal{S})$ следующим образом. Для любой нормализованной системы в форме Сколема $S \in \mathcal{S}$ положим

$$CS(S) = \mathrm{eq}(S).$$

Лемма 2.18. *Функция $CS : \mathcal{S} \to P(\mathcal{S})$ является эффективным не сильно пренебрежимым клонированием.*

Доказательство. Пусть $S_1, S_2 \in \mathcal{D}$ – нормализованные системы в форме Сколема. Пусть $\mathrm{eq}(S_1) \cap \mathrm{eq}(S_2) \neq \varnothing$ и $S \in \mathrm{eq}(S_1) \cap \mathrm{eq}(S_2)$. Тогда система S получается из системы S_1 добавлением уравнений, и из системы S_2 добавлением уравнений. Но это означает, что либо S_1 получается из S_2 добавлением уравнений и тогда $S_1 \in \mathrm{eq}(S_2)$, откуда $\mathrm{eq}(S_1) \subseteq \mathrm{eq}(S_2)$, либо S_2 получается из S_1 добавлением уравнений и тогда $\mathrm{eq}(S_2) \subseteq \mathrm{eq}(S_1)$. Таким образом, функция CS является клонированием.

То, что CS является эффективным клонированием, следует из существования алгоритма, который по любой системе S перечисляет множество $\mathrm{eq}(S)$ в порядке возрастания размеров, а системы одинакового размера в лексикографическом порядке.

То, что клонирование CS не сильно пренебрежимо, следует из леммы 2.17. □

Определим теперь множество

$$SOL_{\mathbb{Z}}(\mathcal{S}) = \{S \in \mathcal{S} : \text{система } S \text{ разрешима над } \mathbb{Z}\}.$$

Теорема 2.9. *Проблема разрешимости нормализованных систем диофантовых уравнений в форме Сколема над множеством целых чисел $(SOL_{\mathbb{Z}}(\mathcal{S}), \mathcal{D})$ сильно неразрешима.*

Доказательство. Заметим, что, если $S \in SOL_{\mathbb{Z}}(\mathcal{S})$, то $\mathrm{eq}(S) \subseteq SOL_{\mathbb{Z}}(\mathcal{S})$. Поэтому для клонирования CS имеет место

$$(CS(SOL_{\mathbb{Z}}(\mathcal{S})), \mathcal{S}) = (SOL_{\mathbb{Z}}(\mathcal{S}), \mathcal{S}).$$

А так, как по лемме 2.18, клонирование CS эффективно и не сильно пренебрежимо, то, по теореме 2.2 проблема $(SOL_{\mathbb{Z}}(\mathcal{S}), \mathcal{D})$ сильно неразрешима. □

2.5 Неразрешимые элементарные теории

Для изучения генерической вычислимости и сложности элементарных теорий необходимо зафиксировать представление замкнутых формул первого порядка. Рассмотрим некоторое естественное представление замкнутых формул языка первого порядка с помощью двоичных деревьев. Это представление, с одной стороны настолько же компактно как и стандартное представление строками символов (с точностью до линейного множителя). С другой стороны, оно удобно для различного рода подсчетов.

Зафиксируем конечную сигнатуру

$$\sigma = \{P_1^{(a_1)}, \ldots, P_k^{(a_k)}, f_1^{(b_1)}, \ldots, f_m^{(b_m)}, c_1, \ldots, c_l\},$$

где P_i предикаты, f_i функции и c_i константы. Положим

$$K = K_\sigma = \max_{i=1,\ldots,k, j=1,\ldots,m} \{a_i, b_j + 1, 2\}.$$

Пусть $\mathfrak{A} = \langle A, \sigma \rangle$ – алгебраическая система сигнатуры σ.

Назовем замкнутую формулу Φ сигнатуры σ *простой атомарной* если она имеет следующий вид:

1) $x_j = f_i(x_{i_1}, \ldots, x_{i_s})$,

2) $P_i(x_{i_1}, \ldots, x_{i_r})$,

3) $x_i = c_j$.

Мы говорим, что замкнутая формула Φ сигнатуры σ имеет *натуральную пренексную* форму, если она имеет вид:

$$\Phi = Q_1 x_1 \ldots Q_t x_t \varphi,$$

где $Q_i \in \{\forall, \exists\}$ – кванторы, φ бескванторная формула, полученная с помощью конъюнкций, дизъюнкций из простых атомарных формул или их отрицаний. Заметим, что любая замкнутая формула может быть приведена с помощью эквивалентных преобразований к натуральной пренексной форме [14]. При этом размер формулы увеличивается не более чем линейно.

Пусть теперь φ – бескванторная формула, которая является булевой комбинацией простых атомарных формул и их отрицаний. Естественным образом можно сопоставить формуле φ бинарное дерево T_φ, которое представляет конструкцию φ из простых атомарных формул и их отрицаний с помощью конъюнкций и дизъюнкций. Внутренние вершины T_φ помечены символами $\vee$ и $\wedge$, а листья T_φ помечены простыми атомарными или их отрицаниями. С другой стороны, по любому такому бинарному дереву можно восстановить бескванторную формулу. Это дает взаимно-однозначное представление бескванторных частей замкнутых формул сигнатуры σ в натуральной пренексной форме размеченными бинарными деревьями. Если T_φ имеет n листьев, то не более Kn переменных могут встретиться в T_φ, поэтому в дальнейшем будем полагать, что все переменные T_φ лежат в множестве $x_1, \ldots, x_{Kn}$.

Пусть $\Phi = Q_1 x_1 \ldots Q_t x_t \varphi$ – формула в натуральной пренексной форме. *Представление* Φ состоит из бинарного дерева T_φ, которое кодирует

бесквантерную часть φ, и кванторной приставки $Q_1x_1 \dots Q_tx_t$. Если T_φ имеет n листьев, то длина кванторной приставки не более Kn. Поэтому число n листьев в дереве T_φ дает линейную верхнюю оценку на число нефиктивных переменных и кванторов в Ф. Заметим также, что число булевых операций в бесквантерной части Ф равно $n-1$. Под размером $size(\Phi)$ формулы Ф будем понимать число n листьев дерева T_φ.

Для упрощения подсчетов будем считать, что формула Ф размера n зависит от всех переменных $\{x_1, \dots, x_{Kn}\}$ и кванторы навешаны на все эти переменные (то есть кванторная приставка содержит ровно Kn кванторов).

Например, вот представление формулы

$$\forall x_1 \exists x_2 \forall x_3 (x_1 = x_2 + x_3) \wedge ((x_2 = x_3) \vee (x_3 \neq x_1))$$

сигнатуры $\{+\}$:

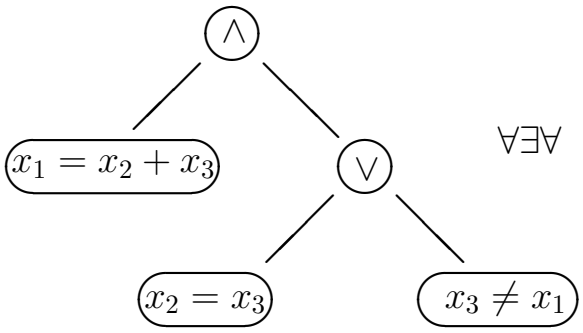

Рисунок 1.

В дальнейшем будем отождествлять замкнутые формулы сигнатуры σ с их представлениями. Кроме того, все формулы будут предполагаться замкнутыми, поэтому слово «замкнутая» будем опускать. Обозначим через $\mathcal{F}$ множество всех формул в натуральной пренексной форме, а через $\mathcal{F}_n$ множество всех формул в $\mathcal{F}$ размера n.

Лемма 2.19. *Число $a(n)$ простых атомарных формул σ от n переменных и их отрицаний равно*

$$a(n) = \sum_{i=1}^{m} 2n^{b_i+1} + \sum_{j=1}^{k} 2n^{a_j} + 2nl.$$

Доказательство. Прямой подсчет. □

Напомним, что числа Каталана C_n определяются следующим образом

$$C_n = \frac{1}{n+1}\binom{2n}{n},$$

где $\binom{2n}{n}$ – соответствующий биномиальный коэффициент. В дальнейшем нам понадобится следующее утверждение о числах Каталана.

Лемма 2.20. *Для $n > m$ имеет место*

$$\frac{C_{n-m}}{C_n} > \frac{1}{4^m}.$$

Доказательство. Оценим отношение чисел Каталана:

$$\frac{C_{n-m}}{C_n} = \frac{n+1}{n-m+1} \times \frac{\binom{2(n-m)}{n-m}}{\binom{2n}{n}} =$$

$$= \frac{n+1}{n-m+1} \times \frac{\frac{(2(n-m))!}{(n-m)!(n-m)!}}{\frac{(2n)!}{n!n!}} > \frac{(2(n-m))!n!n!}{(n-m)!(n-m)!(2n)!} =$$

$$= \frac{n!}{(n-m)!} \times \frac{2(n-m)\dots(n-m+1)}{2n\dots(n+1)} =$$

$$= \frac{n(n-1)\dots(n-m+1)2(n-m)\dots(n-m+1)}{2n\dots(n+1)} =$$

$$= \frac{(n\dots(n-m+1))^2}{2n\dots(2(n-m)+1)} > \Big(\frac{(n-1)\dots(n-m)}{2(n-1)\dots(2n-2m)}\Big)^2 > \frac{1}{2^{2m}} = \frac{1}{4^m}.$$

□

Лемма 2.21. $|\mathcal{F}_n| = 2^{(K+1)n-1} a(Kn)^n C_{n-1}$.

Доказательство. Любая формула из $\mathcal{F}$ размера n состоит из кванторной приставки длины Kn и бинарного дерева с n листьями и $n-1$ внутренними вершинами. Очевидно, что существуют 2^{Kn} различных кванторных приставки длины Kn. Известно (см., например, [7]), что существует C_{n-1} неразмеченных бинарных деревьев с n листьями. Каждая внутренняя вершина такого дерева может быть помечена символами $\vee$ или $\wedge$, поэтому есть всего 2^{n-1} таких разметок. Каждый лист может быть помечен простой атомарной формулой от Kn переменных или ее отрицанием, поэтому существует $a(Kn)^n$ таких разметок. Это показывает, что

$$|\mathcal{F}_n| = 2^{Kn} \times 2^{n-1} \times a(Kn)^n C_{n-1}.$$

□

Пусть $\mathfrak{A} = \langle A, \sigma \rangle$ – алгебраическая система конечной сигнатуры σ. Для любой формулы Φ определим множества

$$AND(\Phi) = \{\Phi \wedge \Psi,\ \Psi \text{ – произвольная формула}\},$$

$$OR(\Phi) = \{\Phi \vee \Psi,\ \Psi \text{ – произвольная формула}\},$$

и множества

$$AND(\Phi)^+ = \{\Phi \wedge \Psi,\ \Psi \text{ – произвольная формула такая, что } \mathfrak{A} \models \Psi\},$$

$$AND(\Phi)^- = \{\Phi \wedge \Psi,\ \Psi \text{ – произвольная формула такая, что } \mathfrak{A} \not\models \Psi\},$$

$$OR(\Phi)^+ = \{\Phi \vee \Psi,\ \Psi \text{ – произвольная формула такая, что } \mathfrak{A} \models \Psi\},$$

$$OR(\Phi)^- = \{\Phi \vee \Psi,\ \Psi \text{ – произвольная формула такая, что } \mathfrak{A} \not\models \Psi\}.$$

Лемма 2.22. *Для любой формулы* Φ *множества* $AND(\Phi)^+$ *и* $OR(\Phi)^-$ *не сильно пренебрежимы. Более того, существует константа* $C > 0$ *такая, что*

$$\frac{|AND(\Phi)^+ \cap \mathcal{F}_n|}{|\mathcal{F}_n|} > \frac{1}{(Cn)^{Km}}$$

для любого $n > m$*, где* m *– размер формулы* Φ*. Аналогичная оценка имеет место и для множества* $OR(\Phi)^-$*.*

Доказательство. Докажем это для множества $AND(\Phi)^+$, для $OR(\Phi)^-$ утверждение доказывается аналогично. Пусть формула Φ имеет размер m. Рассмотрим все формулы вида

$$Q_1 x_1 \dots Q_{Kn} x_{Kn} (\varphi \wedge \psi), \tag{3}$$

где φ – бесквантorная часть формулы Φ от переменных $x_1, \dots, x_{Km}$, а ψ – произвольная бесквантorная формула размера $n - m$ от переменных $x_{Km+1}, \dots, x_{Kn}$. Так как множества переменных формул φ и ψ различны, то любую формулу вида (3) можно записать в виде $\Phi \wedge \Psi \in AND(\Phi)_n$. Таким образом, множество таких формул (обозначим его S) является подмножеством $AND(\Phi)_n$.

Количество формул вида (3) равно количеству всевозможных бесквантorных частей и кванторов для $K(n-m)$ переменных формулы ψ (кванторы для переменных формулы φ зафиксированы). Отсюда

$$|S| = 2^{K(n-m)} \cdot 2^{(n-m)-1} \cdot (a(Kn))^{(n-m)} \cdot C_{n-m-1} =$$

$$= 2^{(K+1)(n-m)-1} \cdot (a(Kn))^{(n-m)} \cdot C_{n-m-1}.$$

Поэтому имеет место оценка

$$\frac{|AND(\Phi)_n|}{|\mathcal{F}_n|} \geq \frac{|S \cap AND(\Phi)_n|}{|\mathcal{F}_n|} =$$

$$= \frac{|S|}{|\mathcal{F}_n|} = \frac{2^{K(n-m+1)-1} \cdot (a(Kn))^{n-m} \cdot C_{n-m-1}}{2^{Kn} \cdot 2^{n-1} \cdot a(Kn)^n C_{n-1}}.$$

Оценим сначала часть без чисел Каталана:

$$\frac{2^{(K+1)(n-m)-1} \cdot (a(Kn))^{n-m}}{2^{Kn} \cdot 2^{n-1} \cdot a(Kn)^n} =$$

$$= \frac{1}{2^{(K+1)m} \cdot a(Kn)^m} > \frac{1}{2^{(K+1)m} \cdot (Bn)^{Km}} = \frac{1}{(2Bn)^{Km}},$$

где константа B уточняется из вида многочлена $a(Kn)$ степени K.

По лемме 2.20 имеем оценку:

$$\frac{C_{n-m-1}}{C_{n-1}} > \frac{1}{4^m}.$$

Таким образом, имеем

$$\frac{|AND(\Phi)_n|}{|\mathcal{F}_n|} > \frac{1}{(C_1 n)^{Km}} \times \frac{1}{4^m} > \frac{1}{(Cn)^{Km}},$$

где $C > 0$ – некоторая константа.

Заметим теперь, что

$$AND(\Phi)_n = AND(\Phi)_n^+ \cup AND(\Phi)_n^-.$$

Действительно, если $\Phi \wedge \Psi \in AND(\Phi)_n$, то $\Phi \wedge \neg\Psi \in AND(\Phi)_n$. В самом деле, если

$$\Psi = Q_1 x_1 \dots Q_{K(n-m)} x_{K(n-m)} \psi(x_1, \dots x_{K(n-m)}),$$

то

$$\neg\Psi = \bar{Q}_1 x_1 \dots \bar{Q}_{K(n-m)} x_{K(n-m)} \neg\psi(x_1, \dots x_{K(n-m)}),$$

где $\bar{\exists} = \forall$, $\bar{\forall} = \exists$. Дерево для $\neg\psi$ можно получить по правилам Де Моргана, заменив $\wedge$ на $\vee$ для внутренних вершин (и $\vee$ на $\wedge$) и, заменив

каждую атомарную формулу листа на ее отрицание. Размер дерева при этом сохранится. Это означает, что для каждой формулы из $AND(\Phi)_n^+$ найдется уникальная формула из $AND(\Phi)_n^-$, и наоборот. Отсюда следует, что

$$|AND(\Phi)_n^+| = |AND(\Phi)_n^-|,$$

поэтому

$$|AND(\Phi)_n^+| = \frac{1}{2}|AND(\Phi)_n|.$$

Откуда и следует утверждение леммы. □

Теорема 2.10. *Пусть $\mathfrak{A} = \langle A, \sigma \rangle$ – алгебраическая система конечной сигнатуры σ с неразрешимой элементарной теорией $Th(\mathfrak{A})$. Проблема распознавания $(Th(\mathfrak{A}), \mathcal{F})$ является сильно неразрешимой.*

Доказательство. Допустим, что проблема распознавания $(Th(\mathfrak{A}), \mathcal{F})$ сильно генерически разрешима. Тогда существует сильно генерический алгоритм $\mathcal{A}$, такой, что множество

$$G(\mathcal{A}) = \{\Phi \in \mathcal{F} : \mathcal{A}(\Phi) \neq ?\}$$

является сильно генерическим, и определяющий для любой замкнутой формулы $\Phi \in G(\mathcal{A})$ истинна она или ложна. Построим теперь алгоритм $\mathcal{B}$, определяющий истинность любой формулы $\Phi \in \mathcal{F}$. Это будет противоречить неразрешимости теории $Th(\mathfrak{A})$. На формуле Φ алгоритм $\mathcal{B}$ будет работать следующим образом

1. Запускает на формуле Φ генерический алгоритм $\mathcal{A}$. Если $\mathcal{A}(\Phi) \neq ?$, то выдает правильный ответ об истинности или ложности формулы Φ. Если $\mathcal{A}(\Phi) = ?$, то переходит к шагу 2.

2. Перебирает формулы из множеств $AND(\Phi)$ и $OR(\Phi)$ до тех пор, пока не получит формулу Ψ такую, что $\mathcal{A}(\Phi) \neq ?$ и с помощью алгоритма $\mathcal{A}$ проверяет истинность Ψ.

3. Если $\Psi = \Phi \wedge \Omega$ и Ψ истинна, то и Φ истинна. Если $\Psi = \Phi \vee \Omega$ и Ψ ложна, то и Φ ложна. В любом другом случае возвращается на шаг 2 и продолжает генерировать формулы.

Докажем, что алгоритм $\mathcal{B}$ остановится на любой формуле Φ. Пусть Φ истинна. По лемме 2.22 множество $AND(\Phi)^+$ не является сильно пренебрежимым, поэтому пересечение $AND(\Phi)^+$ с сильно генерическим множеством $G(\mathcal{A})$ непусто. Это означает, что рано или поздно на шаге 2 алгоритма встретится истинная формула $\Phi \wedge \Omega$ из множества $G(\mathcal{A})$, для которой алгоритм $\mathcal{A}$ определит ее истинность, а с ней и истинность Φ. Аналогично доказывается корректность работы алгоритма на ложной формуле Φ. $\square$

Будем называть формулу $\Phi \in \mathcal{F}$ *нормализованной*, если в дереве T_φ, представляющем бескванторную часть φ формулы Φ, для любой переменной x_i, $i > q$ найдется переменная x_{i-1}, расположенная либо в том же листе дерева, либо в более левом. Другими словами, переменные в листьях дерева T_φ занумерованы слева направо. В представление нормализованной формулы размера n, помимо дерева T_φ и кванторной приставки $Q_1 Q_2 \ldots Q_{Kn}$, будет входить еще перестановка $\pi_\Phi = (i_1, i_2, \ldots, i_{Kn})$ индексов переменных $(1, 2, \ldots, Kn)$, показывающая на какую переменную x_m навешан квантор Q_m. Зачем нужна эта дополнительная информация, будет видно из следующего утверждения.

Лемма 2.23. *Пусть $\mathfrak{A} = \langle A, \sigma \rangle$ – произвольная алгебраическая система конечной сигнатуры σ. Существует алгоритм, который по любой*

замкнутой формуле $\Phi \in \mathcal{F}$ *сигнатуры* σ *строит нормализованную формулу* Φ' *сигнатуры* σ *такую, что* $\mathfrak{A} \models \Phi$ *тогда и только тогда, когда* $\mathfrak{A} \models \Phi'$.

Доказательство. Пусть T_φ – дерево T_φ, представляющее бескванторную часть φ формулы Φ размера n. Перенумеруем переменные $x_1, \ldots, x_{Kn}$ так, чтобы дерево T_φ стало нормализованным. При этом также выполним соответствующую перестановку индексов переменных π_Φ. Например, для формулы

$$\forall x_1 \forall x_2 \exists x_3 (((x_1 = x_3) \vee (x_3 = x_1 + x_2)) \& (x_1 = x_2 x_3)) \vee (x_2 \neq x_3 x_1)$$

получится нормализованная формула

$$\forall x_1 \forall x_3 \exists x_2 (((x_1 = x_2) \vee (x_2 = x_1 + x_3)) \& (x_1 = x_2 x_3)) \vee (x_3 \neq x_2 x_1),$$

а соответствующая перестановка будет $(1, 3, 2)$. Очевидно, что для получившейся нормализованной формулы Φ' имеет место $\mathfrak{A} \models \Phi \Leftrightarrow \mathfrak{A} \models \Phi'$. □

Обозначим через $\mathcal{NF}$ множество всех нормализованных формул сигнатуры σ. Для любой нормализованной формулы $\Phi = Q_1 Q_2 \ldots Q_{Kt} \varphi$ размера t рассмотрим множество нормализованных формул

$$\mathrm{eq}(\Phi) = \big\{ Q'_1 Q'_2 \ldots Q'_{Kn} \big(\varphi \vee ((x_1 \neq x_1) \wedge \psi) \big),\ \pi \big\},$$

где $n > t$ – любое, ψ – произвольная бескванторная нормализованная формула от переменных $x_1, \ldots, x_{Kn}$, π – произвольная перестановка множества индексов $(1, \ldots, Kn)$, а кванторы $Q'_1, \ldots, Q'_{Kn}$ – любые такие, что $Q_i = Q'_{\pi(i)}$ для всех $i = 1, \ldots, Kt$.

Лемма 2.24. *Пусть* $\mathfrak{A} = \langle A, \sigma \rangle$ *– произвольная алгебраическая система конечной сигнатуры* σ*. Для любой нормализованной формулы* Φ *сигнатуры* σ *для любой формулы* $\Phi' \in eq(\Phi)$ *имеет место* $\mathfrak{A} \models \Phi$ *тогда и только тогда, когда* $\mathfrak{A} \models \Phi'$*.*

Доказательство. Утверждение следует непосредственно из определения множества $\mathrm{eq}(\Phi)$. □

Лемма 2.25. *Для любой нормализованной формулы* Φ *множество* $eq(\Phi)$ *не является пренебрежимым.*

Доказательство. Любая нормализованная формула Φ размера n, состоит из Kn кванторов, перестановки π_Φ индексов переменных $(1, 2, \ldots, Kn)$ и бинарного дерева T_φ с n листьями и $n-1$ внутренней вершиной. Существует 2^{Kn} вариантов выбрать кванторы. Существуют C_{n-1} неизоморфных бинарных деревьев с n листьями, где C_{n-1} – это $n-1$-ое число Каталана. Каждая внутренняя вершина может быть помечена либо $\vee$, либо $\wedge$ (всего $n-1$ таких вершин – 2^{n-1} вариантов разметки). Обозначим через L_i число способов разметки i-го листа дерева T_φ (нумерация листов дерева слева направо). Кроме того, существует $(Kn)!$ способов выбора перестановки π_Φ. В итоге получается

$$|\mathcal{NF}_n| = (Kn)! L_1 L_2 \ldots L_n 2^{K(n+1)-1} C_{n-1}.$$

Теперь посчитаем число формул Φ размера n из множества $\mathrm{eq}(\Phi)$. Это делается аналогично, с той лишь разницей, что там у формул произвольной может быть лишь поддерево, отвечающее за формулу ψ. Оно имеет $n-t-1$ листьев, листья могут быть размечены $L_{t+2}, \ldots, L_n$ способами. Перестановка π_Φ может быть любая, а кванторы $Q'_1, \ldots, Q'_{Kn}$ – любые такие, что $Q_i = Q'_{\pi(i)}$ для всех $i = 1, \ldots, Kt$. То есть, из Kn кванторов Kt

зафиксированы, остальные могут быть произвольными. Итого получаем

$$|\mathrm{eq}(\Phi)_n| = (Kn)!L_{t+2}L_2 \dots L_n 2^{K(n-t)-1} C_{n-t-2}.$$

Оценим теперь асимптотическую плотность множества $\mathrm{eq}(\Phi)$:

$$\rho(\mathrm{eq}(\Phi)) = \lim_{n\to\infty} \frac{|\mathrm{eq}(\Phi)_n|}{|\mathcal{NF}_n|} =$$

$$= \lim_{n\to\infty} \frac{(Kn)!L_{t+2}L_2 \dots L_n 2^{K(n-t)-1} C_{n-t-2}}{(Kn)!L_1 L_2 \dots L_n 2^{K(n+1)-1} C_{n-1}} =$$

$$= \lim_{n\to\infty} \frac{C_{n-t-2}}{L_1 L_2 \dots L_{t+1} 2^{K(t+1)} C_{n-1}} =$$

$$= \frac{1}{L_1 L_2 \dots L_{t+1} 2^{K(t+1)}} \lim_{n\to\infty} \frac{C_{n-t-2}}{C_{n-1}} >$$

$$> \frac{1}{L_1 L_2 \dots L_{t+1} 2^{K(t+1)}} \lim_{n\to\infty} \frac{1}{4^{t-1}} = const > 0.$$

Здесь использована лемма 2.20 для оценки отношения чисел Каталана. Также использовано то, что число способов разметить лист L_i, $i \le t = const$, является константой – это следует из нормализованности представления формул. Лемма доказана. □

Определим функцию $CF : \mathcal{NF} \to P(\mathcal{NF})$ следующим образом. Для любой нормализованной формулы $\Phi \in \mathcal{NF}$ положим

$$CF(\Phi) = \mathrm{eq}(\Phi).$$

Лемма 2.26. *Функция $CF : \mathcal{NF} \to P(\mathcal{NF})$ является эффективным не пренебрежимым клонированием.*

Доказательство. Пусть $\Phi_1, \Phi_2 \in \mathcal{NF}$ – нормализованные формулы. Пусть $\mathrm{eq}(\Phi_1) \cap \mathrm{eq}(\Phi_2) \neq \varnothing$ и $\Phi \in \mathrm{eq}(\Phi_1) \cap \mathrm{eq}(\Phi_2)$. Тогда формула Φ содержит и формулу Φ_1 и формулу Φ_2. Но это означает, что либо Φ_1 содержит Φ_2 и тогда $\Phi_2 \in \mathrm{eq}(\Phi_1)$, откуда $\mathrm{eq}(\Phi_1) \subseteq \mathrm{eq}(\Phi_2)$, либо Φ_2 содержит Φ_1 и тогда $\Phi_1 \in \mathrm{eq}(\Phi_2)$, откуда $\mathrm{eq}(\Phi_2) \subseteq \mathrm{eq}(\Phi_1)$. Таким образом, функция CF является клонированием.

То, что CF является эффективным клонированием, следует из существования алгоритма, который по любой нормализованной формуле Φ перечисляет множество $\mathrm{eq}(\Phi)$ в порядке возрастания размеров формул, а формулы одинакового размера в лексикографическом порядке.

То, что клонирование CF не сильно пренебрежимо, следует из леммы 2.25. □

Теорема 2.11. *Пусть $\mathfrak{A} = \langle A, \sigma \rangle$ – алгебраическая система конечной сигнатуры σ с неразрешимой элементарной теорией $Th(\mathfrak{A})$. Проблема распознавания $(Th(\mathfrak{A}), \mathcal{NF})$ является супер неразрешимой.*

Доказательство. Заметим, что по лемме 2.24, если $\Phi \in Th(\mathfrak{A})$, то $\mathrm{eq}(\Phi) \subseteq Th(\mathfrak{A})$. Поэтому для клонирования CF имеет место

$$(CF(Th(\mathfrak{A})), \mathcal{NF}) = (Th(\mathfrak{A}), \mathcal{NF}).$$

А так, как по лемме 2.26, клонирование CF эффективно и непренебрежимо, то, по теореме 2.2 проблема $(Th(\mathfrak{A}), \mathcal{NF})$ является супер неразрешимой. □

2.6 Генерическая теорема Гёделя о неполноте

Напомним, что под формальной арифметикой понимается множество замкнутых формул первого порядка сигнатуры $\sigma_{FA} = \{+, \cdot, 0, 1\}$, которые

выводятся с помощью правил логического вывода из аксиом исчисления предикатов первого порядка с равенством и следующих арифметических аксиом:

1. $\forall x \forall y\ x + y = y + x$,
2. $\forall x\ x + 0 = x$,
3. $\forall x \forall y \forall z\ (x + y) + z = x + (y + z)$,
4. $\forall x \forall y \forall z\ x + z = y + z \to x = y$,
5. $\forall x \forall y\ x \cdot y = y \cdot x$,
6. $\forall x\ x \cdot 0 = 0$,
7. $\forall x\ x \cdot 1 = x$,
8. $\forall x \forall y \forall z\ (x \cdot y) \cdot z = x \cdot (y \cdot z)$,
9. $\forall x \forall y \forall z\ x \cdot (y + z) = x \cdot y + x \cdot z$,
10. $\Phi(0) \wedge \forall x\ (\Phi(x) \to \Phi(x + 1)) \to \forall x \Phi(x)$, где Φ – любая формула сигнатуры σ_{FA}.

В этом подразделе будем использовать представление формул сигнатуры $\sigma_{FA} = \{+, \cdot, 0, 1\}$, введенное в подразделе 2.5.

Теорема 2.12. *Пусть формальная арифметика непротиворечива. Имеет место следующее:*

1. *Не существует сильно генерического множества арифметических формул $\mathcal{G} \subseteq \mathcal{F}$ такого, что для любой формулы $\Phi \in \mathcal{G}$ либо Φ, либо $\neg\Phi$ выводится из аксиом формальной арифметики.*

2. *Не существует генерического множества арифметических формул $\mathcal{G} \subseteq \mathcal{NF}$ такого, что для любой формулы $\Phi \in \mathcal{G}$ либо Φ, либо $\neg\Phi$ выводится из аксиом формальной арифметики.*

Доказательство. Докажем пункт 1. Допустим противное, то есть, что существует такое сильно генерическое множество формул $\mathcal{G} \subseteq \mathcal{F}$. По теореме Гёделя о неполноте, существует такая арифметическая формула Φ, что ни Φ, ни $\neg\Phi$ невыводимы из аксиом формальной арифметики. Рассмотрим множества $AND(\Phi)^+$ и $AND(\neg\Phi)^+$. По лемме 2.22 эти множества не сильно пренебрежимы, а, значит $AND(\Phi)^+ \cap \mathcal{G} \neq \varnothing$ и $AND(\neg\Phi)^+ \cap \mathcal{G} \neq \varnothing$. Таким образом, существуют истинные формулы Ψ_1 и Ψ_2 такие, что $\Phi \wedge \Psi_1 \in \mathcal{G}$ и $\neg\Phi \wedge \Psi_2 \in \mathcal{G}$. В множестве $\mathcal{G}$ любая формула или ее отрицание доказуемы, поэтому одна из формул

$$\Phi \wedge \Psi_1,\ \neg\Phi \vee \neg\Psi_1,$$

доказуема, и одна из формул

$$\neg\Phi \wedge \Psi_2,\ \Phi \vee \neg\Psi_2,$$

доказуема. Обе формулы $\neg\Phi \vee \neg\Psi_1$ и $\Phi \vee \neg\Psi_2$ не могут быть доказуемы, так как одна из них обязательно будет ложной (напомним, что Ψ_1 и Ψ_2 истины). Поэтому либо $\Phi \wedge \Psi_1$, либо $\neg\Phi \wedge \Psi_2$ доказуема. Отсюда следует, что либо Φ, либо $\neg\Phi$ доказуема, а это противоречит теореме Гёделя. Пункт 1 доказан.

Докажем пункт 2. Пусть $\mathcal{G} \subseteq \mathcal{NF}$ – любое генерическое подмножество арифметических формул. По теореме Гёделя о неполноте, существует такая арифметическая формула Φ, что ни Φ, ни $\neg\Phi$ не выводимы из аксиом формальной арифметики. Заметим, что все формулы из $\mathrm{eq}(\Phi)$ эквивалентны Φ в том смысле, что они выводимы или не выводимы в

формальной арифметике одновременно с Φ. Заметим также, что если $\Psi \in \mathrm{eq}(\Phi)$, то $\neg\Psi \in \mathrm{eq}(\neg\Phi)$. По лемме 2.25 множество $\mathrm{eq}(\Phi)$ не является пренебрежимым, поэтому $eq(\Phi) \cap \mathcal{G} \neq \varnothing$. Значит найдется формула $\Psi \in eq(\Phi) \cap \mathcal{G}$. Кроме того, $\neg\Psi \in eq(\neg\Phi)$. Теперь, если Ψ выводима в формальной арифметике, то и Φ выводима. А если $\neg\Psi$ выводима, то и $\neg\Phi$ выводима. В любом случае получаем противоречие с теоремой Гёделя. Значит ни Ψ, ни $\neg\Psi$ невыводимы из аксиом формальной арифметики. Пункт 2 доказан. □

2.7 Ограничения генерической амплификации

Возникает естественный вопрос: любую ли генерически неразрешимую проблему можно получить из подходящей проблемы, неразрешимой в классическом смысле, с помощью метода генерической амплификации? В данном подразделе дается негативный ответ на этот вопрос: доказывается, что любое простое пренебрежимое множество не является генерически разрешимым и, в то же время не может быть построено с помощью метода амплификации из какого-либо рекурсивно перечислимого множества. С другой стороны, доказывается, что любое рекурсивно перечислимое множество с ненулевой асимптотической плотностью может быть получено с помощью генерической амплификации из множества натуральных чисел. В конце подраздела строится пример так называемого абсолютно неразрешимого множества, которое остается неразрешимым на любом непренебрежимом подмножестве входов.

Лемма 2.27. *Любое простое негенерическое множество $A \subseteq \omega$ супер неразрешимо.*

Доказательство. Допустим, что простое негенерическое множество $A \subseteq$

ω генерически разрешимо с помощью некоторого генерического алгоритма $\mathcal{A}$. Тогда множество $\{x \in \omega : \mathcal{A}(x) = 0\}$ является непренебрежимым, а, следовательно, и бесконечным рекурсивно перечислимым подмножеством иммунного множества $\overline{A}$. Противоречие. □

Лемма 2.28. *Существует простое пренебрежимое множество.*

Доказательство. Пусть $S_1, S_2, S_3, \ldots$ – эффективная нумерация всех рекурсивно перечислимых множеств. Будем строить рекурсивно перечислимое множество S так, чтобы для каждого бесконечного рекурсивно перечислимого множества S_i оно содержало какой-либо его элемент. Это будет гарантировать, что в $\omega \setminus S$ ни одно из бесконечных S_i не содержится. Кроме того будем следить, чтобы S получилось пренебрежимым. Отсюда $\omega \setminus S$ будет иммунным, а S простым.

Опишем процесс эффективного перечисления S по шагам. Вначале S пусто. На i-м шаге ищем элемент из S_i, размер которого не совпадает с размерами ранее добавленных в S элементов. Если такой элемент найдется (в случае бесконечного S_i он обязательно найдется), добавляем его в S.

В результате получится множество S, которое пересекается с любым бесконечным рекурсивно перечислимым множеством, то есть простое множество. Кроме того, для каждого размера n (длины двоичной записи числа) S содержит ровно один элемент размера n. Это следует из процесса построения S, а также того, что мы добавляем элементы из рекурсивно перечислимых множеств $T_k = \{2^k, 2^{k+1}, \ldots\}$, которые содержат элементы всевозможных размеров. Поэтому

$$\rho(S) = \lim_{n\to\infty} \frac{|S \cap \omega_n|}{|\omega_n|} = \lim_{n\to\infty} \frac{1}{2^n} = 0.$$

То есть S пренебрежимо. □

Лемма 2.29. *Пусть $A \subseteq \omega$ – непустое пренебрежимое множество. Не существует непренебрежимого клонирования $C : \omega \to P(\omega)$ такого, что $A = C(S)$ для некоторого множества $S \subseteq \omega$.*

Доказательство. Допустим, что существуют множество $S \subseteq \omega$ и непренебрежимое клонирование $C : \omega \to P(\omega)$ такие, что $A = C(S)$. Тогда, для любого $s_0 \in S$ множество $C(s_0)$ непренебрежимо и $C(s_0) \subseteq A$, что противоречит пренебрежимости A. □

Следующая теорема дает пример генерически неразрешимого рекурсивно перечислимого множества, которое нельзя получить с помощью метода генерической амплификации.

Теорема 2.13. *Пусть $A \subseteq \omega$ – простое пренебрежимое множество. Тогда*

1. *Множество A супер неразрешимо.*
2. *Не существует непренебрежимого клонирования $C : \omega \to P(\omega)$ такого, что $A = C(S)$ для некоторого множества $S \subseteq \omega$.*

Доказательство. Пункты 1, 2 следуют из лемм 2.27 и 2.29. Существование простых пренебрежимых множеств было доказано в лемме 2.28. □

С другой стороны, любое рекурсивно перечислимое множество натуральных чисел с ненулевой асимптотической плотностью можно всегда получить из множества ω с помощью генерической амплификации.

Теорема 2.14. *Пусть $A \subseteq \omega$ – любое рекурсивно перечислимое множество с ненулевой асимптотической плотностью. Тогда существует эффективное непренебрежимое клонирование $C : \omega \to P(\omega)$ такое, что $A = C(\omega)$.*

Доказательство. Пусть $A \subseteq \omega$ – рекурсивно перечислимое множество и $\rho(A) > \varepsilon > 0$, где ε – рациональное число. По основной теореме о рекурсивно перечислимых множествах, существует всюду определенная инъективная вычислимая функция $f : \omega \to \omega$ такая, что множество A совпадает с множеством значений f. То есть

$$A = \{f(0), f(1), f(2), \ldots, f(n), \ldots\}.$$

Обозначим через $A^{(i)}$ множество, состоящее из первых i элементов A, перечисленных функцией f, то есть

$$A^{(i)} = \{f(0), f(1), \ldots, f(i)\}.$$

Напомним, что множество $A_k^{(i)}$ состоит из всех элементов $A^{(i)}$ размера k.

Определим вычислимую функцию $E : \omega \times \omega \to \omega$, следующим образом. Чтобы вычислить $E(i, j)$ при $i > 0$, перебираем значения $f(0), f(1), \ldots$ до тех пор, пока $j + 1$ раз не найдем такое $f(k)$, что

$$\sum_{s=1}^{i-1} \frac{\varepsilon}{2^s} < \rho_m(A_m^{(k)}) \leq \sum_{s=1}^{i} \frac{\varepsilon}{2^s},$$

где $m = size(f(k))$. В случае $i = 1$ левую границу полагаем равной 0. Положим $E(i, j) = f(k)$. Чтобы вычислить $E(0, j)$ перебираем значения $f(0), f(1), \ldots$ до тех пор, пока $j + 1$ раз не найдем такое $f(k)$, что

$$\rho_m(A_m^{(k)}) \geq \varepsilon,$$

где $m = size(f(k))$. Положим $E(0, j) = f(k)$.

Определим теперь клонирование $C : \omega \to P(\omega)$ следующим образом. Для любого натурального x положим

$$C(x) = \{E(x, n) : n \in \omega\}.$$

Из построения функции E видно, что клонирование эффективно и $A = C(\omega)$. Также из построения функции E следует, что

$$\rho(C(0)) \geq \rho(A) - \varepsilon > 0$$

и

$$\rho(C(n)) \geq \frac{\varepsilon}{2^n}$$

при $n > 0$. Таким образом, клонирование C непренебрежимо. □

Будем называть множество $A \subseteq \omega$ *абсолютно неразрешимым*, если не существует алгоритма $\mathcal{A} : \omega \to \{0, 1, ?\}$ такого, что

1. $\forall x \in \omega\ \mathcal{A}(x)\downarrow$,
2. множество $\{x \in \omega : \mathcal{A}(x) \neq ?\}$ непренебрежимо,
3. $\mathcal{A}(x) \neq ? \Rightarrow \mathcal{A}(x) = \chi_A(x)$, где $\chi_A(x)$ – характеристическая функция множества A.

Теорема 2.15. *Пусть $A \subseteq \omega$ – простое пренебрежимое множество. Тогда A абсолютно неразрешимо.*

Доказательство. Допустим, существует алгоритм $\mathcal{A} : \omega \to \{0, 1, ?\}$, который останавливается для всех входов, такой, что множество

$$\{x \in \omega : \mathcal{A}(x) \neq ?\}$$

непренебрежимо, и такой, что

$$\mathcal{A}(x) \neq ? \ \Rightarrow\ \mathcal{A}(x) = \chi_A(x).$$

Но тогда рекурсивно перечислимое множество

$$\{x \in \omega : \mathcal{A}(x) = 0\}$$

есть непренебрежимое, а стало быть, бесконечное подмножество иммунного множества $\overline{A}$. Противоречие. Значит, такого алгоритма не существует, поэтому множество A абсолютно неразрешимо. □

Отметим, что Бенвеню, Дей и Хольц в [30], используя коды Адамара, доказали, что в каждой ненулевой тьюринговой степени существуют абсолютно неразрешимые множества.

3 Генерически трудноразрешимые алгоритмические проблемы

3.1 Арифметика Пресбургера

Напомним, что арифметика Пресбургера – это теория первого порядка алгебраической системы $\mathfrak{N} = \langle \mathbb{N}, \sigma_{PA} \rangle$ с сигнатурой $\sigma_{PA} = \{+, 1\}$. В этом подразделе будем использовать представление формул сигнатуры σ_{PA}, введенное в подразделе 2.5.

Теорема 3.1. *Не существует сильно генерического экспоненциального алгоритма, решающего* $(Th(\mathfrak{N}), \mathcal{F})$.

Доказательство. Пусть существует сильно генерический экспоненциальный алгоритм $\mathcal{A}$, решающий арифметику Пресбургера. Тогда множество

$$G(\mathcal{A}) = \{\Phi \in \mathcal{F} : \mathcal{A}(\Phi) \neq ?\}$$

является сильно генерическим.

Построим экспоненциальный алгоритм $\mathfrak{B}$, который будет определять истинность любой формулы. Это будет противоречить теореме Рабина-Фишера. На формуле Φ размера n алгоритм $\mathcal{B}$ будет работать следующим образом.

1. Запускает на формуле Φ генерический алгоритм $\mathcal{A}$. Если $\mathcal{A}(\Phi) \neq ?$, то выдает правильный ответ об истинности или ложности формулы Φ. Если $\mathcal{A}(\Phi) = ?$, то переходит к шагу 2.

2. Перебирает формулы из множеств $AND(\Phi)$ и $OR(\Phi)$ размера $\leq n^2$ до тех пор, пока не получит формулу Ψ такую, что $\mathcal{A}(\Phi) \neq ?$ и с помощью алгоритма $\mathcal{A}$ проверяет истинность Ψ.

3. Если $\Psi = \Phi \wedge \Omega$ и Ψ истинна, то и Φ истинна. Если $\Psi = \Phi \vee \Omega$ и Ψ ложна, то и Φ ложна. В любом другом случае возвращается на шаг 2 и продолжает генерировать формулы.

Нужно показать, что алгоритм $\mathfrak{B}$ будет работать за экспоненциальное время. Необходимо оценить время работы алгоритма $\mathfrak{B}$ на входе Φ. Допустим Φ истинна (для ложной формулы доказательство аналогично). Покажем, что на шаге 2 формула из множества $AND(\Phi)^+$, которая попадает в генерическое множество G имеет размер не более n^2. По лемме 2.21 всего формул размера от n до n^2 не более

$$|\mathcal{F}_n| + \ldots + |\mathcal{F}_{n^2}| < n^2|\mathcal{F}_{n^2}| < n^2 \cdot 2^{3n^2-1} n^{3n^2} C_{n-1} < 2^{p(n)},$$

где $p(n)$ – некоторый полином. Поэтому и весь алгоритм будет в целом работать за время $2^{q(n)}$, где $q(n)$ – полином.

Покажем, что пересечение $AND(\Phi)^+_{n^2}$ и $G(\mathcal{A})_{n^2}$ непусто. Так как, $G(\mathcal{A})$ сильно генерическое, то существуют константы $D > 0$ и $\alpha > 0$ такие, что

$$\frac{|(\mathcal{F} \setminus G(\mathcal{A}))_{n^2}|}{|\mathcal{F}_{n^2}|} < \frac{D}{2^{\alpha n^2}}$$

для любого n. По лемме 2.22 существует константа $C > 0$ такая, что

$$\frac{|AND(\Phi)^+_{n^2}|}{|\mathcal{F}_{n^2}|} > \frac{1}{(Cn)^{4n}}.$$

При достаточно большом n имеет место

$$\frac{D}{2^{\alpha n^2}} < \frac{1}{(Cn)^{4n}}.$$

Откуда

$$\frac{|(\mathcal{F} \setminus G(\mathcal{A}))_{n^2}|}{|\mathcal{F}_{n^2}|} < \frac{|AND(\Phi)^+_{n^2}|}{|\mathcal{F}_{n^2}|},$$

поэтому

$$|(\mathcal{F} \setminus G(\mathcal{A}))_{n^2}| < |AND(\Phi)^+_{n^2}|,$$

и в множестве $AND(\Phi)^+_{n^2}$ должны быть формулы из $G(\mathcal{A})$. $\square$

3.2 Проблема выполнимости булевых формул

Классическое представление булевых формул с помощью таблиц истинности с практической точки зрения является громоздким в том смысле, что размер таблицы истинности растет экспоненциально с ростом числа переменных. Гораздо более компактным и практичным является представление формул с помощью бинарных деревьев. Оно часто используется в программировании различных приложений, связанных с символьными вычислениями. Кроме того, оно удобно для для различного рода подсчетов.

Пусть φ – булева формула в базисе $\{\vee, \wedge, \neg\}$. Без ограничения общности можно считать, что в ней отрицания находятся только над переменными. Любую булеву формулу можно легко привести к такому виду с помощью законов Де Моргана, поэтому в дальнейшем будем рассматривать только такие формулы. Естественным образом можно сопоставить формуле φ бинарное дерево T_φ, которое представляет конструкцию φ из переменных и их отрицаний с помощью конъюнкций и дизъюнкций. Внутренние вершины T_φ помечены символами $\vee$ и $\wedge$, а листья T_φ помечены переменными или их отрицаниями. С другой стороны, по любому такому бинарному дереву можно восстановить булеву формулу. Это дает взаимно-однозначное представление булевых формул размеченными бинарными деревьями. Если T_φ имеет n листьев, то не более n переменных могут встретиться в T_φ, поэтому в дальнейшем будем полагать, что все переменные T_φ лежат в множестве $x_1, \ldots, x_n$. *Представление* φ состоит из бинарного дерева T_φ. Заметим также, что число булевых операций в бесквантворной части φ равно $n - 1$. Под размером $size(\varphi)$ формулы φ будем понимать число листьев n.

Например, вот представление формулы $(\neg x_1) \wedge (x_2 \vee \neg x_3)$:

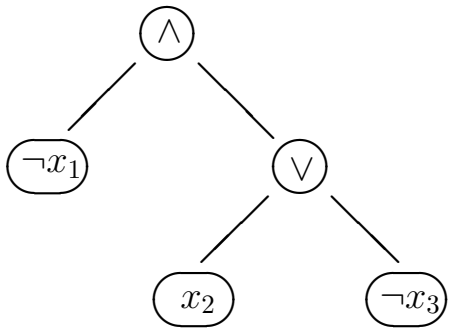

Рисунок 2.

В дальнейшем будем отождествлять булевы формулы с их представлениями. Обозначим через $\mathcal{BF}$ множество всех булевых формул, а через $\mathcal{BF}_n$ множество всех формул в $\mathcal{BF}$ размера n.

Лемма 3.1. $|\mathcal{BF}_n| = 2^{n-1}(2n)^n C_{n-1}$.

Доказательство. Любая формула из $\mathcal{BF}$ размера n есть размеченное бинарное дерево с n листьями и $n-1$ внутренними вершинами. Известно (см., например, [7]), что существует C_{n-1} неразмеченных бинарных деревьев с n листьями. Каждая внутренняя вершина такого дерева может быть помечена символами $\vee$ или $\wedge$, поэтому есть всего 2^{n-1} таких разметок. Каждый лист может быть помечен одной из n переменных или ее отрицанием, поэтому существует $(2n)^n$ таких разметок. Это показывает, что

$$|\mathcal{BF}_n| = 2^{n-1}(2n)^n C_{n-1}.$$

□

Для любой формулы φ определим множество

$$AND(\varphi) = \{\varphi \wedge \psi,\ \psi - \text{произвольная булева формула}\}.$$

Лемма 3.2. *Для любой формулы φ множество $AND(\phi)$ не сильно пренебрежимо. Более того*

$$\frac{|AND(\varphi) \cap \mathcal{BF}_n|}{|\mathcal{BF}_n|} > \frac{1}{(16n)^k}$$

для любого $n > k$, где k – размер формулы φ.

Доказательство. Пусть формула φ имеет размер k. Тогда для любой формулы $\varphi \wedge \psi$ из множества $AND(\varphi) \cap \mathcal{BF}_n$ формула ψ должна иметь размер $n-k$. Кроме того, в этой формуле может участвовать любая из n переменных. Поэтому, аналогично тому как это делалось в доказательстве леммы 3.1, можно подсчитать

$$|AND(\varphi) \cap \mathcal{BF}_n| = 2^{n-k-1}(2n)^{n-k}C_{n-k-1}.$$

Отсюда

$$\frac{|AND(\varphi) \cap \mathcal{F}_n|}{|\mathcal{F}_n|} = \frac{2^{n-k-1}(2n)^{n-k}C_{n-k-1}}{2^{n-1}(2n)^n C_{n-1}} =$$

$$= \frac{1}{(4n)^k} \times \frac{C_{n-k-1}}{C_{n-1}} > \frac{1}{(4n)^k} \times \frac{1}{4^k} = \frac{1}{(16n)^k}.$$

Здесь мы использовали лемму 2.20 для оценки отношения чисел Каталана. Таким образом, имеем

$$\frac{|AND(\Phi)_n|}{|\mathcal{F}_n|} > \frac{1}{(16n)^k},$$

что и требовалось доказать. □

Теорема 3.2. *Если существует сильно генерический полиномиальный алгоритм, решающий проблему выполнимости на множестве $\mathcal{BF}$, то существует вероятностный полиномиальный алгоритм, решающий эту проблему для всего множества $\mathcal{BF}$.*

Доказательство. Допустим, что существует сильно генерический полиномиальный алгоритм $\mathcal{A}$, решающий проблему выполнимости на множестве $\mathcal{BF}$. Тогда множество

$$G(\mathcal{A}) = \{\varphi \in \mathcal{BF} : \mathcal{A}(\varphi) \neq ?\}$$

является сильно генерическим.

Построим вероятностный полиномиальный алгоритм $\mathcal{B}$, определяющий выполнимость любой формулы φ. На формуле φ размера n алгоритм $\mathcal{B}$ будет работать следующим образом

1. Вычисляет $\mathcal{A}(\varphi)$. Если $\mathcal{A}(\varphi) \neq ?$, то с помощью алгоритма $\mathcal{A}$ определяет выполнимость φ. Если $\mathcal{A}(\varphi) = ?$, то переходит к шагу 2.

2. Генерирует случайную формулу ψ размера $n^2 - n$.

3. Вычисляет $\mathcal{A}(\varphi \wedge \psi)$ и $\mathcal{A}(\varphi \wedge \neg\psi)$. Если оба значения неравны «?», то с помощью алгоритма $\mathcal{A}$ определяет их выполнимость и переходит к шагу 4. Если нет, то выдает ответ 0.

4. Так как обязательно, хотя бы одна из формул ψ и $\neg\psi$ выполнима, то возможны следующие варианты:

   - Если $\varphi \wedge \psi$ и $\varphi \wedge \neg\psi$ невыполнимы, то φ тоже невыполнима.
   - Если хотя бы одна из них выполнима, то и φ тоже выполнима.

   В любом из этих двух случаев алгоритм выдает правильный ответ.

Заметим, что алгоритм выдает правильный ответ на шагах 1 и 4, а на шаге 3 может выдать неправильный ответ. Нужно доказать, что вероятность того, что ответ выдается на шаге 3, меньше $1/2$.

Поэтому вероятность того, что случайная формула вида $\varphi \wedge \psi$ из $AND(\varphi)_{n^2}$ не попадет в $G(\mathcal{A})$ не больше

$$\frac{|(\mathcal{BF} \setminus G(\mathcal{A}))_{n^2}|}{|AND(\varphi)_{n^2}|} = \frac{|(\mathcal{BF} \setminus G(\mathcal{A}))_{n^2}|}{|\mathcal{BF}_{n^2}|} \times \frac{|\mathcal{BF}_{n^2}|}{|AND(\Phi)_{n^2}|}.$$

Так как, $G(\mathcal{A})$ сильно генерическое, то существует константа $\alpha > 0$ такая, что

$$\frac{|(\mathcal{BF} \setminus G(\mathcal{A}))_{n^2}|}{|\mathcal{BF}_{n^2}|} < \frac{1}{2^{\alpha n^2}}$$

для любого n. С другой стороны, по лемме 3.2

$$\frac{|\mathcal{BF}_{n^2}|}{|AND(\Phi)_{n^2}|} < (16n^2)^n.$$

Поэтому искомая вероятность не больше

$$\frac{(16n^2)^n}{2^{\alpha n^2}} = \frac{2^{4n+2n\log n}}{2^{\alpha n^2}}$$

и при больших n меньше $1/4$. Аналогично делается оценка для формул вида $\varphi \wedge \neg\psi$. Вероятность же непопадания в $G(\mathcal{A})$ хотя бы одной из формул $\varphi \wedge \psi$ или $\varphi \wedge \neg\psi$ не больше $1/4 + 1/4 = 1/2$. Это и означает то, что вероятность выдачи ответа на шаге 3 меньше $1/2$.

Осталось доказать полиномиальность алгоритма. Для этого нужно за полиномиальное время уметь генерировать случайно и равномерно формулу размера $N = n^2 - n$. Это делается следующим образом.

1. Генерируем некоторую последовательность (далее «слово») из N символов a и $N - 1$ символов p.

2. Делаем такой циклический сдвиг этого слова, чтобы оно начиналось на символ a и заканчивалось на p. Этому слову соответствует обратная польская запись для скобочного выражения от символов a.

3. По слову ищем скобочное выражение, следующим образом: пробегаем по всем символам слова, если встречаем символ a, то помещаем его в стек. Если встречаем символ p, то извлекаем 2 элемента из стека, затем добавляем между ними символ p, заключаем их в скобки и помещаем в стек. Если по ходу процедуры стек окажется пуст, то переходим к шагу 2. Если все пройдет нормально, и мы дойдем до конца слова и при этом в стеке останется всего 1 элемент, то искомым скобочным выражением и будет этот элемент. Иначе – переходим к шагу 2.

4. Вместо букв p подставляем $\vee$ или $\wedge$ - равновероятно.

5. Каждую букву a в слове заменяем переменную $x_1, \ldots, x_N$ или ее отрицание (какую - выбираем равновероятно).

Корректность этого алгоритма и равномерность генерации формул следует из того, что существует взаимно-однозначное соответствие между обратными польскими записями из N символов a и $N-1$ символов p и бинарными деревьями с N листьями, которые помечены символом a (см. [7]).

Итак, в предположении существования полиномиального сильно генерического множества, на котором проблема выполнимости булевых формул разрешима за полиномиальное время, мы построили вероятностный полиномиальный алгоритм, разрешающий эту проблему на всем множестве формул. □

Доказанную теорему можно переформулировать следующим образом.

Теорема 3.3. *Если $P \neq NP$ и $P = BPP$, то не существует сильно генерического полиномиального алгоритма, решающего проблему выпол-*

нимости на множестве $\mathcal{BF}$.

3.3 Проблема распознавания квадратичных вычетов

Пусть $\mathbb{Z}/(m)$ – мультипликативная группа вычетов по модулю $m \in \mathbb{N}$. *Квадратичным вычетом* в группе $\mathbb{Z}/(m)$ называется любой элемент x, для которого существует $y \in \mathbb{Z}/(m)$ такой, что $x = y^2$. В противном случае элемент x называется *квадратичным невычетом*. Рассмотрим следующее множество входов:

$$RES = \{(m, x) \in \mathbb{N}^2 : m = pq, \text{ где } p, q - \text{простые числа}, x \in \mathbb{Z}/(m)\}.$$

Под *проблемой распознавания квадратичных вычетов* понимается проблема распознавания (QR, RES), где

$$QR = \{(m, x) \in RES : x - \text{ квадратичный вычет в } \mathbb{Z}/(m)\}.$$

В настоящее время неизвестно полиномиальных алгоритмов (в том числе и вероятностных), решающих проблему распознавания квадратичных вычетов для всех таких модулей m. Более того, на предположении о ее трудноразрешимости основаны некоторые криптографические алгоритмы (см. [17]).

Для изучения генерической сложности проблемы распознавания квадратичных вычетов необходимо провести некоторую стратификацию на множестве входов. Рассмотрим любую бесконечную последовательность натуральных чисел $\mu = \{m_1, m_2, \ldots\}$ такую, что для любого $n > 1$ имеет место $2^n < m_n < 2^{n+1}$ и m_n – произведение двух различных простых чисел. Будем называть такую последовательность *экспоненциальной*. Из знаменитого постулата Бертрана, доказанного П.Л.Чебышевым, следует, что экспоненциальные последовательности существуют. Теперь

определим алгоритмическую проблему $(QR(\mu), RES(\mu))$ как ограничение проблемы распознавания квадратичных вычетов (QR, RES) на следующее множество входных данных:

$$RES(\mu) = \{(m, x) : m \in \mu, x \in \mathbb{Z}/(m)\}.$$

Под размером входа (m, x) понимается количество бит в двоичной записи числа m минус 1. Заметим, что множество $RES(\mu)_n$ входов проблемы $(QR(\mu), RES(\mu))$ размера n состоит из всех пар (m, x), где m – единственное число $m \in \mu$, удовлетворяющее условию $2^n < m < 2^{n+1}$, а x – любой элемент из $\mathbb{Z}/(m)$.

Таким образом, проблема $(QR(\mu), RES(\mu))$ является подпроблемой проблемы (QR, RES). Тем не менее, можно доказать, что среди проблем $(QR(\mu), RES(\mu))$ существуют проблемы, такие же сложные, как и оригинальная проблема (QR, RES).

Лемма 3.3. *Если не существует полиномиального вероятностного алгоритма для проблемы (QR, RES), то найдется такая экспоненциальная последовательность μ, что и для решения проблемы $(QR(\mu), RES(\mu))$ не существует полиномиального вероятностного алгоритма.*

Доказательство. Пусть $P_1, P_2, \ldots$ – все полиномиальные вероятностные алгоритмы. Из предположения о том, что не существует полиномиального вероятностного алгоритма для проблемы (QR, RES) следует, что для любого алгоритма P_n существует бесконечно много групп $\mathbb{Z}/(m)$, в которых он не может решить (QR, RES). Из этого следует, что можно выбрать последовательность натуральных чисел $\mu' = \{m_1, m_2, \ldots\}$, являющихся произведениями двух простых, так, чтобы алгоритм P_n не решал

(QR, RES) в группе $\mathbb{Z}/(m_n)$, и для любого n выполнялось $m_{n+1} > 2m_n$. Последовательность μ' можно расширить до экспоненциальной последовательности μ , добавив где нужно новые члены. Заметим теперь, что $(QR(\mu), RES(\mu))$ и будет той проблемой, для которой не существует полиномиального вероятностного алгоритма. □

Теорема 3.4. *Если проблема $(QR(\mu), RES(\mu))$ генерически полиномиально разрешима, то существует полиномиальный вероятностный алгоритм, решающий $(QR(\mu), RES(\mu))$ для всех входов.*

Доказательство. Допустим, существует генерический полиномиальный алгоритм $\mathcal{A}$, разрешающий проблему $(QR(\mu), RES(\mu))$. Тогда множество

$$G(\mathcal{A}) = \{(m, x) : \mathcal{A}(m, x) \neq ?\}$$

является генерическим.

Построим вероятностный полиномиальный алгоритм $\mathcal{B}$, решающий $(QR(\mu), RES(\mu))$ на всем множестве входов. Алгоритм $\mathcal{B}$ на входе (m, x) будет работать следующим образом:

1. Вычисляет $\mathcal{A}(m, x)$. Если $\mathcal{A}(m, x) \neq ?$, то выдает правильный ответ. Если нет, переходит к шагу 2.

2. Генерирует случайно и равновероятно элемент $y \in \mathbb{Z}/(m)$. Вычисляет $z = xy^2$.

3. Вычисляет $\mathcal{A}(m, z)$.

4. Если $\mathcal{A}(m, z) \neq ?$ то определяет, является ли z квадратичным вычетом. Очевидно, что $z = xy^2$ является квадратичным вычетом тогда и только тогда, когда таковым является x. Выдает правильный ответ.

5. Если $\mathcal{A}(m, z) =?$, выдает ответ 0.

Заметим, что полиномиальный вероятностный алгоритм $\mathcal{B}$ может выдать неправильный ответ только на шаге 5. Докажем, что вероятность этого меньше $1/2$. Для этого заметим, что в группе $\mathbb{Z}/(m)$, где $m = pq$ с простыми p и q, ровно $1/4$ элементов являются квадратами, а потому $z = xy^2$ может равновероятно принимать $1/4$ значений в $\mathbb{Z}/(m)$. В то же время с ростом размера входа для более $3/4$ элементов $u \in \mathbb{Z}/(m)$ входы (m, u) попадают в генерическое множество G. Поэтому вероятность шага 5 стремится к 0 с ростом размера входа и становится меньше $1/2$. $\square$

Непосредственным следствием доказанной теоремы является следующее утверждение.

Следствие 3.1. *Если для проблемы (QR, RES) не существует полиномиального вероятностного алгоритма, то существует экспоненциальная последовательность μ такая, что проблема $(QR(\mu), RES(\mu))$ не является генерически полиномиально разрешимой.*

3.4 Проблема дискретного логарифма

Проблема дискретного логарифма состоит в вычислении функции $dl : I \to \mathbb{N}$, где I – это множество троек (a, p, g_p) таких, что p – простое число, g_p – фиксированный первообразный элемент в поле $GF(p)$ и $a \in GF(p)$, $a \neq 0$. Сама функция dl определяется следующим образом:

$$dl(a, p, g_p) = x \Leftrightarrow g_p^x = a \in GF(p).$$

Под размером входа понимается число разрядов в двоичной записи числа p. В настоящее время неизвестно полиномиальных алгоритмов (даже

вероятностных), решающих проблему дискретного логарифма. Это обстоятельство лежит в основе криптостойкости многочисленных криптографических алгоритмов ([17]).

Для изучения генерической сложности этой проблемы дискретного логарифма необходимо провести некоторую стратификацию на множестве входов. Рассмотрим любую бесконечную последовательность простых чисел $\pi = \{p_1, p_2, \ldots, p_n, \ldots\}$, удовлетворяющую условию $2^n \leq p_n < 2^{n+1}$ для любого $n > 0$. Такие последовательности существуют благодаря известному постулату Бертрана, доказанному П.Л.Чебышевым. Будем называть такую последовательность *экспоненциальной*. Теперь определим функцию dl_π как ограничение функции dl на множество троек (a, p, g_p) таких, что $p \in \pi$. Заметим, что для этой функции множество всех входов размера n состоит из троек (a, p, g_p) с фиксированными p, g_p и произвольным $a \in [1, p-1]$. Очевидно, что проблема вычисления dl_π является подпроблемой вычисления dl. Следующая лемма показывает, что некоторые такие подпроблемы так же трудны как и оригинальная проблема дискретного логарифма.

Лемма 3.4. *Если не существует полиномиального вероятностного алгоритма для вычисления dl, то найдется такая экспоненциальная последовательность простых чисел π, что и для вычисления dl_π нет полиномиального вероятностного алгоритма.*

Доказательство. Пусть $P_1, P_2, \ldots$ – все полиномиальные вероятностные алгоритмы. Из предположения о том, что не существует полиномиального вероятностного алгоритма для вычисления dl следует, что для любого алгоритма P_n существует бесконечно много полей $GF(p)$, в которых он не может вычислить dl. Из этого следует, что можно выбрать последо-

вательность $\pi' = \{p_1, p_2, \ldots\}$ так, чтобы алгоритм P_n не вычислял dl в поле $GF(p_n)$ и для любого n выполнялось бы $p_{n+1} > 2p_n$. Последовательность π' можно расширить до экспоненциальной последовательности π, добавив где нужно новые члены. Заметим теперь, что dl_π и будет той функцией, для вычисления которой не существует полиномиального алгоритма. □

Следующий результат говорит о том, что проблема дискретного логарифма остается вычислительно трудной и в генерическом случае, при условии ее трудноразрешимости в худшем.

Теорема 3.5. *Пусть π – любая экспоненциальная последовательность простых чисел. Если существует полиномиальный генерический алгоритм, вычисляющий функцию dl_π, то существует полиномиальный вероятностный алгоритм, вычисляющий dl_π для всех входов.*

Доказательство. Пусть существует полиномиальный генерический алгоритм $\mathcal{A}$, вычисляющий функцию dl_π. Построим вероятностный полиномиальный алгоритм $\mathcal{B}$, вычисляющий dl_π на всем множестве входов. Алгоритм $\mathcal{B}$ на входе (a, p, g_p) будет работать следующим образом.

1. Сгенерировать случайно и равномерно $y \in [0, \ldots, p-1]$ и вычислить $a' = ag_p^y$.
2. Запустить алгоритм $\mathcal{A}$ на (a', p, g_p).
3. Если $\mathcal{A}(a', p, g_p) = z \in \mathbb{N}$, то $a' = g_p^z = ag_p^y = g_p^{x+y}$, откуда $x = z - y \mod (p-1)$ – дискретный логарифм для исходной задачи (a, p, g_p).
4. Если $\mathcal{A}(a', p, g_p) =?$, то выдать 0.

Заметим, что данный полиномиальный вероятностный алгоритм может выдать неправильный ответ только на шаге 4. Докажем, что вероятность этого меньше $1/2$. Действительно, $a' = ag_p^y$ при $y \in [0, \ldots, p-1]$ пробегает все ненулевые элементы поля $GF(p)$, поэтому множество

$$\{(a', p, g_p) \ : \ y \in [0, \ldots, p-1]\}$$

совпадает со множеством всех входов размера n. Но алгоритм $\mathcal{A}$ генерический, поэтому доля тех входов (a', p, g_p), на которых он выдает неопределенный ответ стремится к 0 с ростом n и, с некоторого момента, становится меньше $1/2$. □

Непосредственным следствием доказанной теоремы является следующее утверждение.

Следствие 3.2. *Если для вычисления функции dl не существует полиномиального вероятностного алгоритма, то существует экспоненциальная последовательность π такая, что для вычисления функции dl_π не существует генерического полиномиального алгоритма.*

3.5 Проблема извлечения корня в группах вычетов

Проблема извлечения корня в группах вычетов состоит в вычислении функции $root : I \to \mathbb{N}$, где I – это множество троек (a, e, m) таких, что $m = pq$ и p, q – различные простые числа, $e < m$, $(\varphi(m), e) = 1$ и $a \in \mathbb{Z}/(m)$. Сама функция $root$ определяется следующим образом:

$$root(a, e, m) = x \Leftrightarrow x^e = a \in \mathbb{Z}/(m).$$

Под размером входа понимается число разрядов в двоичной записи числа m. Заметим, что при условии $(\varphi(m), e) = 1$ для любого $a \in \mathbb{Z}/(m)$

существует единственный корень степени e (см. [8]). Этим корнем является элемент a^d, где $d = e^{-1} \mod \varphi(m)$. В настоящее время неизвестно полиномиальных алгоритмов (даже вероятностных), решающих проблему извлечения корня в группах вычетов. На этом факте, в частности, основана криптостойкость известной криптосистемы RSA (см. [76, 17]).

Для изучения генерической сложности проблемы извлечения корня в группах вычетов необходимо провести стратификацию на множестве входов. Рассмотрим любую бесконечную последовательность пар натуральных чисел $\mu = \{(e_1, m_1), (e_2, m_2), \ldots\}$ такую, что для любого $n > 1$ имеет место $2^n < m_n < 2^{n+1}$, m_n – произведение двух различных простых чисел, и $e_n < m_n$ и $(\varphi(m_n), e_n) = 1$. Будем называть такую последовательность *экспоненциальной*. Из знаменитого постулата Бертрана, доказанного П.Л.Чебышевым, следует, что экспоненциальные последовательности существуют.

Теперь определим функцию $root_\mu$ как ограничение функции $root$ на следующее множество входных данных:

$$I = \{(a, e, m) : (e, m) \in \mu, a \in \mathbb{Z}/(m)\}.$$

Под размером входа (a, e, m) понимается количество бит в двоичной записи числа m. Заметим, что множество I_n входов функции $root_\mu$ размера n состоит из всех пар (a, e, m), где (e, m) – единственная пара из μ, удовлетворяющая условию $2^n < m < 2^{n+1}$, а a – любой элемент из $\mathbb{Z}/(m)$. Таким образом, проблема вычисления функции $root_\mu$ является подпроблемой проблемы проблемы вычисления функции $root$. Следующая лемма доказывает, что среди функций $root_\mu$ существуют функции, такие же сложные, как и оригинальная функция $root$.

Лемма 3.5. *Если не существует полиномиального вероятностного алгоритма для вычисления функции root, то найдется такая экспоненциальная последовательность μ, что и для вычисления $root_\mu$ нет полиномиального вероятностного алгоритма.*

Доказательство. Пусть $P_1, P_2, \ldots$ – все полиномиальные вероятностные алгоритмы. Из предположения о том, что не существует полиномиального вероятностного алгоритма для вычисления $root$ следует, что для любого алгоритма P_n существует бесконечно много групп $\mathbb{Z}/(m)$, в которых он не может вычислить функцию $root$ для некоторых степеней e. Из этого следует, что можно выбрать последовательность $\mu' = \{(e_1, m_1), (e_2, m_2), \ldots\}$ так, чтобы алгоритм P_n не вычислял $root$ в группе $\mathbb{Z}/(m_n)$ со степенью e_n и для любого n выполнялось бы $m_{n+1} > 2m_n$. Последовательность μ' можно расширить до экспоненциальной последовательности μ, добавив где нужно новые члены. Заметим теперь, что $root_\mu$ и будет той функцией, для вычисления которой не существует полиномиального алгоритма. □

Следующий результат говорит о том, что проблема извлечения корня в группах вычетов остается вычислительно трудной и в генерическом случае, при условии ее трудноразрешимости в худшем случае.

Теорема 3.6. *Пусть μ – любая экспоненциальная последовательность. Если существует полиномиальный генерический алгоритм, вычисляющий функцию $root_\mu$, то существует полиномиальный вероятностный алгоритм, вычисляющий $root_\mu$ для всех входов.*

Доказательство. Пусть существует полиномиальный генерический алгоритм $\mathcal{A}$, вычисляющий функцию $root_\mu$. Построим вероятностный поли-

номиальный алгоритм $\mathcal{B}$, вычисляющий $root_\mu$ на всем множестве входов. Алгоритм $\mathcal{B}$ на входе (a, e, m) будет работать следующим образом.

1. Сгенерировать случайно и равномерно натуральное $b < m$.
2. Если $(b, m) = 1$, то положить $a' = ab^e$.
   (a) Запустить алгоритм $\mathcal{A}$ на (a', e, m).
   (b) Если $\mathcal{A}(a', e, m) = y \in \mathbb{Z}/(m)$, то $a' = y^e = ab^e$, откуда $a = (yb^{-1})^e$ и корень e-й степени из a равен yb^{-1}.
   (c) Если $\mathcal{A}(a', e, m) =?$, то выдать 1.
3. Если $(b, m) \neq 1$, то $(b, m) = p$, где p – один из двух делителей числа $m = pq$. Таким образом можно легко посчитать $\varphi(m) = (p-1)(q-1)$ и найти корень как элемент a^d, где $d = e^{-1} \mod \varphi(m)$.

Заметим, что данный полиномиальный вероятностный алгоритм может выдать неправильный ответ только на шаге 2в. Докажем, что вероятность этого меньше $1/2$. Действительно, $a' = ab^e$ при $b \in \mathbb{Z}/(m)$ и $(b, m) = 1$ пробегает все элементы группы $\mathbb{Z}/(m)$, поэтому множество

$$\{(ab^e, e, m) \ : \ b \in \mathbb{Z}/(m), \ (b, m) = 1\}$$

совпадает со множеством всех входов размера n. Но алгоритм $\mathcal{A}$ генерический, поэтому доля тех входов (a', e, m), на которых он выдает неопределенный ответ стремится к 0 с ростом n и, с некоторого момента становится меньше $1/2$. □

Непосредственным следствием доказанной теоремы является следующее утверждение.

Следствие 3.3. *Если для вычисления функции root не существует полиномиального вероятностного алгоритма, то существует экспоненциальная последовательность μ такая, что для вычисления функции $root_\mu$ не существует генерического полиномиального алгоритма.*

3.6 Проблема поиска изоморфизма графов

Напомним, что два графа G_1 и G_2 называются изоморфными, если существует биекция π между множествами вершин G_1 и G_2 такая, что для любых вершин v, u графа G_1 v и u соединены ребром в G_1 тогда и только тогда, когда $\pi(v)$ и $\pi(u)$ соединены ребром в G_2. Биекция π, осуществляющая изоморфизм, является перестановкой множества вершин графов G_1, G_2, если вершины обоих графов занумерованы числами $\{1, 2, \ldots, n\}$. Классическая проблема изоморфизма графов – это проблема вычисления следующей функции, определенной на множестве всех пар графов с одинаковым числом вершин.

$$gi(G_1, G_2) = \begin{cases} 1, & \text{если графы } G_1, G_2 \text{ изоморфны,} \\ 0, & \text{иначе.} \end{cases}$$

Графы можно представлять с помощью матриц смежности, при этом размер входа – это число вершин n. До сих пор неизвестно, существует ли полиномиальный алгоритм, решающий эту проблему в худшем случае. Но существует много полиномиальных алгоритмов, решающих ее для почти всех входов [28, 63].

Множеством входов для проблемы поиска изоморфизма графов является множество всех пар изоморфных графов. В этой проблеме требуется вычислять следующую функцию.

$$sgi(G_1, G_2) = \text{перестановка, осуществляющая изоморфизм } \pi$$

между изоморфными графами G_1 и G_2.

Также, как и для проблемы изоморфизма графов, до сих пор неизвестно полиномиальных алгоритмов для вычисления sgi в худшем случае. Неизвестно даже вероятностных алгоритмов. Поэтому существуют приложения этой проблемы в криптографии [32].

Для изучения генерической сложности этой проблемы необходимо провести стратификацию на множестве входов. Рассмотрим бесконечную последовательность графов $\gamma = \{G_1, G_2, \ldots, G_n, \ldots\}$ такую, что G_n имеет n вершин для любого n. Для каждой последовательности графов γ определим функцию sgi_γ – ограничение функции sgi на множество пар (G, G_n), где $G_n \in \gamma$ и граф G изоморфен графу G_n. Заметим, что множество всех входов размера n в проблеме вычисления функции sgi_γ состоит из всевозможных пар (G, G_n) с произвольным графом G, изоморфным фиксированному графу G_n из последовательности γ. Очевидно, что проблема вычисления функции sgi_γ является подпроблемой проблемы поиска изоморфизма графов. Следующее утверждение говорит о том, что проблема вычисления функции sgi_γ для некоторых последовательностей γ может быть также трудна, как и проблема вычисления функции sgi.

Лемма 3.6. *Если не существует полиномиального вероятностного алгоритма для вычисления функции sgi, то найдется последовательность графов γ такая, что не существует полиномиального вероятностного алгоритма для вычисления функции sgi_γ.*

Доказательство. Пусть $P_1, P_2, \ldots$ – все полиномиальные вероятностные алгоритмы. Если не существует полиномиального вероятностного алгоритма для вычисления функции sgi, то для любого вероятностного полиномиального алгоритма P_n найдется бесконечно много пар изоморф-

ных графов, для которых P_n не может вычислить sgi. Из этого следует, что можно выбрать последовательность $\gamma' = \{G_1, G_2, \ldots, G_n, \ldots\}$ такую, что алгоритм P_n не может вычислить sgi для (H_n, G_n) для всех n. Более того, γ' упорядочена по возрастанию числа вершин в графах. Теперь можно расширить последовательность γ' до последовательности графов γ с графами G_n для всех размеров n. Из построения γ следует, что не существует полиномиального вероятностного алгоритма для вычисления функции sgi_γ. □

Теорема 3.7. *Если существует полиномиальный генерический алгоритм для вычисления функции sgi_γ, то существует вероятностный полиномиальный алгоритм для вычисления функции sgi_γ на всех входах.*

Доказательство. Предположим, что существует полиномиальный генерический алгоритм $\mathcal{A}$, вычисляющий sgi_γ. Построим вероятностный полиномиальный алгоритм $\mathcal{B}$, вычисляющий sgi_γ для всех входов. Алгоритм $\mathcal{B}$ работает на входе (G, G_n) следующим образом.

1. Запускает алгоритм $\mathcal{A}$ на (G, G_n).
2. Если $\mathcal{A}(G, G_n) \neq ?$, то $\mathcal{B}$ выдает ответ $\mathcal{A}(G, G_n)$ и останавливается, иначе идет на шаг 3.
3. Генерирует случайно и равномерно перестановку φ на вершинах $\{1, \ldots, n\}$ и вычисляет граф $G' = \varphi(G)$.
4. Запускает алгоритм $\mathcal{A}$ на (G', G_n).
5. Если $\mathcal{A}(G', G_n) = ?$ выдает тождественную перестановку (неправильный ответ).

6. Если $\mathcal{A}(G', G_n) = \tau$ – изоморфизм между G' и G_n, то $\tau(G') = \tau(\varphi(G)) = G_n$, откуда изоморфизм между G и G_n есть $\pi = \tau\varphi$.

Для доказательства корректности работы вероятностного алгоритма, надо показать, что вероятность того, что $\mathcal{A}(G', G_n) =?$ меньше $1/2$. Заметим, что $\varphi(G)$ пробегает все множество графов, изоморфных G_n. Множество

$$\{(G, G_n) : \mathcal{A}(G', G_n) =?\}$$

пренебрежимо, поэтому вероятность того, что $\mathcal{A}(G', G_n) =?$ стремится к 0 при увеличении n. □

Непосредственным следствием доказанной теоремы является следующее утверждение.

Следствие 3.4. *Если для вычисления функции sgi не существует полиномиального вероятностного алгоритма, то существует последовательность графов γ такая, что для вычисления функции sgi_γ не существует генерического полиномиального алгоритма.*

4 Генерические сводимости

В данном разделе будем предполагать, что натуральные числа ω представляются в двоичном виде, а размер натурального числа n есть длина его двоичной записи $[\log_2 n] + 1$.

4.1 Генерическая m-сводимость

Множество $A \subseteq \omega$ *генерически m-сводится* к $B \subseteq \omega$ (обозначается $A \leq_{gm} B$), если существует вычислимая функция $f : \omega \to \omega$ такая, что

1. $\forall x \in \omega\ x \in A \Leftrightarrow f(x) \in B$,
2. $\forall S \subseteq \omega\ S$ непренебрежимо $\Rightarrow f(S)$ непренебрежимо.

Как обычно, $A \equiv_{gm} B$ означает, что $A \leq_{gm} B$ и $B \leq_{gm} A$. Также будем писать $A <_{gm} B$, если $A \leq_{gm} B$ и $B \not\leq_{gm} A$. Определим gm-степень множества A как $d_{gm}(A) = \{B \subseteq \omega : B \equiv_{gm} A\}$.

Предложение 4.1. *Для любых множеств $A, B, C \subseteq \omega$ выполнено*

1. $A \leq_{gm} A$.
2. $A \leq_{gm} B \Rightarrow A \leq_m B$.
3. $A \leq_{gm} B$, $B \leq_{gm} C \Rightarrow A \leq_{gm} C$.
4. *Если $A \leq_{gm} B$ и B генерически разрешимо, то A генерически разрешимо.*

Доказательство. Пункты (1) и (2) очевидны.

Для доказательства (3) допустим, что $A \leq_{gm} B$ с помощью вычислимой функции f и $B \leq_{gm} C$ с помощью вычислимой функции g. Тогда

$A \leq_{gm} C$ с помощью $g(f)$, так как для любого $S \subseteq \omega$ множество S непренебрежимо $\Rightarrow$ $f(S)$ непренебрежимо $\Rightarrow$ $g(f(S))$ непренебрежимо.

Для того, чтобы доказать (4), предположим, что $A \leq_{gm} B$ с помощью вычислимой функции f. Так как B генерически разрешимо, то существует генерический алгоритм $\mathcal{B}$, вычисляющий характеристическую функцию множества B. Тогда алгоритм $\mathcal{A} = \mathcal{B}(f(x))$ будет генерическим алгоритмом, вычисляющим характеристическую функцию χ_A множества A. Докажем, что множество $\{x \in \omega : \mathcal{A}(x) =?\}$ пренебрежимо. Действительно, предположим противное, что

$$S = \{x \in \omega : \mathcal{A}(x) =?\} = \{x \in \omega : \mathcal{B}(f(x)) =?\}$$

непренебрежимо. Тогда, по определению генерической m-сводимости, $f(S)$ тоже непренебрежимо. Но

$$f(S) = \{y \in \omega : \exists x \ f(x) = y \text{ и } \mathcal{B}(y) =?\} \subseteq \{x \in \omega : \mathcal{B}(x) =?\}.$$

Получается противоречие: непренебрежимое множество $f(S)$ является подмножеством пренебрежимого множества $\{x \in \omega : \mathcal{B}(x) =?\}$. Из определения m-сводимости следует, что если $\mathcal{A}(x) \neq?$, то $\mathcal{A}(x) = \chi_A(x)$. $\square$

Лемма 4.1. *Пусть $A \leq_1 B$ с помощью вычислимой монотонно возрастающей функции $f(x)$ такой, что существует константа $C > 0$ такая, что $f(x) < Cx$ для всех $x > 0$. Тогда $A \leq_{gm} B$.*

Доказательство. Пусть $S \subseteq \omega$ – непренебрежимое множество. Тогда существует константа $\varepsilon > 0$ такая, что для любого натурального N найдется $n > N$ такое, что

$$\rho_n(S) = \frac{|\{x : |x| = n, x \in S\}|}{2^n} > \varepsilon.$$

Нам нужно доказать, что $f(S)$ непренебрежимо. Так как $f(x) < Cx$, то

$$|f(x)| \leq |x| + C_1 = n + C_1,$$

где $C_1 > 0$ – константа. Так как f монотонно возрастающая, то размеры элементов множества $f(S)$ попадают в отрезок $[n, n + C_1]$. Значит, существует $n' \in [n, n + C_1]$ такое, что

$$|\{f(x) : |f(x)| = n', f(x) \in f(S)\}| \geq \frac{|\{x : |x| = n, x \in S\}|}{C_1}.$$

Теперь

$$\rho_{n'}(f(S)) = \frac{|\{f(x) : |f(x)| = n', f(x) \in f(S)\}|}{2^{n'}} \geq$$

$$\geq \frac{|\{x : |x| = n, x \in S\}|}{C_1 2^n 2^{C_1}} = \frac{\rho_n(S)}{C_1 2^{C_1}} > \frac{\varepsilon}{C_1 2^{C_1}}.$$

Из этого следует, что $f(S)$ непренебрежимо. □

Напомним, что сочленением множеств $A \subseteq \omega$ и $B \subseteq \omega$ называется множество $A \oplus B = \{2a : a \in A\} \cup \{2b + 1 : b \in A\}$.

Лемма 4.2. *Для любых множеств $A, B \subseteq \omega$ имеет место $A \leq_{gm} A \oplus B$ и $B \leq_{gm} A \oplus B$.*

Доказательство. По лемме 4.1 множество A gm-сводится к $A \oplus B$ с помощью функции $f(x) = 2x$, а множество B gm-сводится к $A \oplus B$ с помощью функции $f(x) = 2x + 1$. □

Предложение 4.2. *Пусть $A, B \subseteq \omega$ – рекурсивные множества, $A, B \neq \varnothing, \omega$, и существуют $\rho(A), \rho(B)$.*

1. *Если $\rho(A) = \rho(B) = 1$, то $A \equiv_{gm} B$.*

2. *Если $\rho(A) = \rho(B) = 0$, то $A \equiv_{gm} B$.*

3. Если $\rho(A), \rho(B) \neq 0$ *и* $\rho(A), \rho(B) \neq 1$, *то* $A \equiv_{gm} B$.

Доказательство. (1) Легко видеть, что $A \leq_{gm} B$ с помощью следующей функции

$$f(x) = \begin{cases} x, \text{ if } x \in A \text{ and } x \in B, \\ x, \text{ if } x \notin A \text{ and } x \notin B, \\ b, \text{ if } x \in A \text{ and } x \notin B, \\ a, \text{ if } x \notin A \text{ and } x \in B, \end{cases}$$

где $a \in A$ и $b \in B$. Аналогично доказывается, что $B \leq_{gm} A$.

Пункт (2) следует из (1).

Рассмотрим пункт (3). Пусть

$$A = \{a_1(i) : i \in \omega\},\ \overline{A} = \{a_2(i) : i \in \omega\},$$

$$B = \{b_1(i) : i \in \omega\},\ \overline{B} = \{b_2(i) : i \in \omega\}$$

– эффективные нумерации множеств в возрастающем порядке. Так как A и B рекурсивные, не конечные и не ко-конечные, то $A \leq_1 B$ с помощью вычислимой функции f, которая переводит $a_1(i)$ в $b_1(i)$ и $a_2(i)$ в $b_2(i)$. Так как $\rho(B) > 0$, то существует константа $C > 0$ такая, что для достаточно больших n имеет место

$$\rho_n(B) = \frac{|B_n|}{2^{n-1}} = \frac{|\{b_1(k) : 2^{n-1} \leq b_1(k) < 2^n\}|}{2^{n-1}} > C.$$

Выберем теперь m максимальное такое, что $2^{n-1} \leq b_1(m) < 2^n$. Теперь

$$C < \frac{|\{b_1(k) : 2^{n-1} \leq b_1(k) < 2^n\}|}{2^{n-1}} \leq \frac{2m}{b_1(m)}.$$

Отсюда мы можем оценить $b_1(m) < \frac{2}{C}m$. То, что оценка $b_1(n) < \frac{2}{C}n$ верна для произвольного достаточно большого n, следует из монотонности функции $b_1()$. Из того, что функция $a_1()$ монотонно возрастает следует, что $n < a_1(n)$ для любого n. Из этих двух неравенств следует, что $b_1(n) < C_1 a_1(n)$ для некоторой константы $C_1 > 0$. Аналогично

мы можем оценить $b_2(n) < C_2 a_2(n)$ с некоторой константой $C_2 > 0$. Теперь из леммы 4.1 следует, что $A \leq_{gm} B$. Аналогично доказывается, что $B \leq_{gm} A$. □

Следующая теорема утверждает, что существуют несравнимые относительно $\leq_{gm}$ рекурсивные множества, лежащие в одной 1-степени.

Предложение 4.3. *Существуют рекурсивные множества A и B такие, что $A \equiv_1 B$, но $A \not\leq_{gm} B$ и $B \not\leq_{gm} A$.*

Доказательство. Пусть A – любое рекурсивное пренебрежимое бесконечное множество и B – любое рекурсивное генерическое множество, не являющееся ко-конечным. Очевидно, $A \equiv_1 B$. Но если предположить, что $A \leq_{gm} B$ с помощью какой-либо вычислимой функции f, то f переводит генерическое множество $\overline{A}$ в пренебрежимое множество $\overline{B}$, что противоречит определению генерической m-сводимости. Обратно, если предположить, что $B \leq_{gm} A$ с помощью некоторой вычислимой функции g, то g переводит генерическое множество B в пренебрежимое множество A, что опять противоречит определению генерической m-сводимости. □

Как и в классическом случае, рекурсивно перечислимое множество S называется gm-полным, если для любого рекурсивно перечислимого множества A имеет место $A \leq_{gm} S$.

Теорема 4.1. *Существует gm-полное рекурсивно перечислимое множество.*

Доказательство. Рассмотрим эффективную нумерацию

$$S_0, S_1, \ldots, S_k, \ldots$$

всех рекурсивно перечислимых множеств. Определим следующее рекурсивно перечислимое множество

$$C = \{2^k(2m+1) : m \in S_k, k = 0, 1, \ldots\}.$$

Теперь из леммы 4.1 следует, что $S_k \leq_{gm} C$ с помощью функции $f_k(x) = 2^k(2x+1)$. □

Следующее утверждение показывает, что существуют несравнимые супер неразрешимые рекурсивно перечислимые множества.

Предложение 4.4. *Существуют супер неразрешимые рекурсивно перечислимые множества $A, B \subseteq \omega$ такие, что $A \not\leq_{gm} B$ и $B \not\leq_{gm} A$.*

Доказательство. Пусть S – пренебрежимое простое множество. Существование таких можеств было установлено в лемме 2.28. По лемме 2.27 множество S супер неразрешимо. Рассмотрим следующие множества:

$$A = S \cup \{2n : n \in \omega\}$$

и

$$B = 2S = \{2x : x \in S\}.$$

Ясно, что множество A тоже простое, непренебрежимое и негенерическое. По лемме 2.27 множество A супер неразрешимо. Из леммы 4.1 следует, что $S \leq_{gm} B$ с помощью функции $f(x) = 2x$. Поэтому множество B также супер неразрешимо. Теперь докажем, что множества A и B несравнимы относительно порядка $\leq_{gm}$.

Допустим, что $A \leq_{gm} B$ с помощью некоторой вычислимой функции f. Тогда $f(A) \subseteq B$. Но B пренебрежимо, так как S пренебрежимо. Поэтому $f(A)$ пренебрежимо. Но A непренебрежимо, что противоречит определению генерической m-сводимости.

Обратно, допустим, что $B \leq_{gm} A$ с помощью некоторой вычислимой функции g. Заметим, что непренебрежимое рекурсивно перечислимое множество $C = \{2k+1 : k \in \omega\}$ является подмножеством множества $\overline{B}$. Таким образом, рекурсивно перечислимое множество $f(C)$ является подмножеством $\overline{A}$. Но $\overline{A}$ иммунно, поэтому $f(C)$ конечно и потому пренебрежимо. Снова имеем противоречие с определением генерической m-сводимости. □

Следующая теорема утверждает, что существуют рекурсивно перечислимые m-степени, содержащие более одной gm-степени.

Предложение 4.5. *Существуют супер неразрешимые рекурсивно перечислимые множества A и B такие, что $A \equiv_m B$, но $A <_{gm} B$.*

Доказательство. Пусть S – пренебрежимое простое множество. Рассмотрим множества $A = S$ и $B = 2S = \{2x : x \in S\}$. По лемме 2.27 множество A супер неразрешимо. $A \leq_{gm} B$ с помощью функции $f(x) = 2x$. Поэтому множество B тоже супер неразрешимо. Более того, $B \leq_m A$ с помощью следующей функции

$$f(x) = \begin{cases} x/2, \text{ если } x \text{ четное}, \\ c, \text{ если } x \text{ нечетное}, \end{cases}$$

где $c \notin S$. Но $B \not\leq_{gm} A$ потому что непренебрежимое рекурсивно перечислимое множество $\{2k+1 : k \in \omega\}$ содержится в $\overline{B}$ а иммунное множество $\overline{A}$ не содержит каких-либо бесконечных рекурсивно перечислимых подмножеств. Поэтому $A <_{gm} B$. □

4.2 Генерическая клонирующая сводимость

Множество $A \subseteq \omega$ *генерически клонированно сводится* к множеству $B \subseteq \omega$ (обозначается это $A \leq_{gc} B$), если либо $A = B$, либо существует такая

всюду определенная вычислимая функция $f : \omega \times \omega \to \omega \cup \{?\}$, что

1. $\forall x \in \omega$ если $f(x,0) \neq ?$, то $\forall n \in \omega\ f(x,n) \neq ?$.
2. Множество $\{x \in \omega : f(x,0) = ?\}$ пренебрежимо.
3. $\forall x \in \omega$ если $f(x,0) \neq ?$, то множество $\{f(x,n) : n \in \omega\}$ непренебрежимо.
4. $\forall x \in \omega$ если $f(x,0) \neq ?$, то
   - $x \in A \Rightarrow \forall n \in \omega\ f(x,n) \in B$.
   - $x \notin A \Rightarrow \forall n \in \omega\ f(x,n) \notin B$.

Рассмотрим подробнее каждое из свойств данного определения. Функция f (если ее значение не равно «?») каждому входу $x \in \omega$ проблемы A сопоставляет непренебрежимое подмножество (свойство 3) входов ω проблемы B. Свойство 4 гарантирует, что это подмножество полностью попадает в B, если $x \in A$, и полностью лежит вне B для $x \notin A$. Свойства 1 и 2 говорят о том, что функция f для пренебрежимого множества входов из ω может выдавать неопределенный ответ.

В отличие от классической m-сводимости, где делается один запрос о принадлежности элемента $f(x)$ второму множеству, при наличии генерической клонирующей сводимости можно делать потенциально неограниченное число запросов о принадлежности элементов $f(x,n)$ второму множеству, причем эти элементы образуют достаточно большое (непренебрежимое) подмножество. Эта сводимость очень похожа на механизм клонирования, описанный в подразделе 2.1.

Предложение 4.6. *Если $A \leq_{gc} B$ и B генерически разрешимо, то и A генерически разрешимо.*

Доказательство. Пусть f – функция, осуществляющая генерическую клонирующую сводимость $A \leq_{gc} B$. Пусть также $\mathcal{B}$ – генерический алгоритм, вычисляющий характеристическую функцию множества B. Генерический алгоритм $\mathcal{A}$, вычисляющий характеристическую функцию множества A работает на входе x следующим образом. Сначала вычисляется $f(x,0)$. Если $f(x,0) =?$, то выдаем «?». Иначе вычисляем значения

$$f(x,0), f(x,1), f(x,2), \ldots$$

до тех пор пока не получим такое n, что $\mathcal{B}(f(x,n)) \neq?$. Это всегда произойдет, так как, с одной стороны множество $\{y \in \omega : \mathcal{B}(y) \neq?\}$ генерическое, а с другой стороны множество $\{f(x,n) : n \in \omega\}$ непренебрежимо. Поэтому

$$\{y \in \omega : \mathcal{B}(y) \neq?\} \cap \{f(x,n) : n \in \omega\} \neq \varnothing.$$

По свойству 4 из определения генерической клонирующей сводимости $\mathcal{B}(f(x,n))$ и будет правильным значением характеристической функции множества A на x. Осталось показать, что множество $\{x \in \omega : \mathcal{A}(x) =?\}$ пренебрежимо. Это следует из свойства 2 генерической клонирующей сводимости $\leq_{gc}$. □

Докажем свойство транзитивности генерической клонирующей сводимости.

Предложение 4.7. *Если $A \leq_{gc} B$ и $B \leq_{gc} C$, то $A \leq_{gc} C$.*

Доказательство. Пусть функция f осуществляет генерическую клонирующую сводимость A к B, а функция g – генерическую клонирующую сводимость B к C. Построим всюду определенную вычислимую функцию h, осуществляющую генерическую клонирующую сводимость A к

C. Значение $h(x,n)$ вычисляется следующим образом. Если $f(x,0) =?$, то и $h(x,n) =?$ для любого $n \in \omega$. Если $f(x,0) \neq?$, то опять вычисляем $f(x,0), f(x,1), \ldots$ до тех пор, пока не найдем такое $k \in \omega$, что $g(f(x,k),0) \neq?$. То, что это произойдет доказывается абсолютно также как и в доказательстве предыдущей теоремы. Теперь полагаем $h(x,n) = g(f(x,k),n)$. Легко проверить, что построенная функция h осуществляет генерическую клонирующую сводимость множества A к множеству C. □

Следующее утверждение показывает, что существуют рекурсивно перечислимые множества, к которым генерически клонированно не сводится никакое другое множество. В частности, это означает, что для этой сводимости существуют минимальные генерически неразрешимые рекурсивно перечислимые степени.

Предложение 4.8. *Имеет место следующее:*

1. *Существует такое супер неразрешимое рекурсивно перечислимое множество $A \subseteq \omega$, что для любого множества $B \neq A$ имеет место $B \not\leq_{gc} A$.*

2. *Существуют рекурсивно перечислимые множества $A, B \subseteq \omega$ такие, что $A \not\leq_{gc} B$ и $B \not\leq_{gc} A$.*

Доказательство. Докажем пункт 1. В качестве множества A можно взять любое пренебрежимое простое множество натуральных чисел. Существование таких множеств было доказано в лемме 2.28. По лемме 2.27, множество A супер неразрешимо. Допустим, что для какого-то множества B имеет место $B \leq_{gc} A$ и f – вычислимая функция, осуществляющая генерическую клонирующую сводимость. Заметим, что не существу-

ет такого $x \in B$, что $f(x,0) \neq ?$. Действительно, тогда непренебрежимое множество $\{f(x,n) : n \in \omega\}$ является подмножеством пренебрежимого множества A, чего не может быть. С другой стороны, не существует и такого $x \notin B$, что $f(x,0) \neq ?$. В этом случае иммунное множество $\overline{A}$ содержит бесконечное рекурсивно перечислимое множество $\{f(x,n) : n \in \omega\}$. Поэтому $f(x,0) = ?$ для любого x. Но это противоречит свойствам 1 и 2 определения генерической клонирующей сводимости. Поэтому $B \not\leq_{gc} A$.

Для доказательства пункта 2 выберем в качестве множеств A и B неравные пренебрежимые простые множества. Из пункта 1 следует, что $A \not\leq_{gc} B$ и $B \not\leq_{gc} A$. □

Напомним конструкцию, предложенную Джокушем и Шуппом в [55]. Для любого $k \in \omega$ определим следующее множество

$$R_k = \{m \in \omega : 2^k \mid m,\ 2^{k+1} \nmid m\}.$$

В двоичной записи числа из R_k имеют нули в k младших разрядах, единицу в $k+1$-м разряде и произвольные цифры в остальных. Легко видеть, что при $n > k+1$

$$\rho_n(R_k) = \frac{2^{n-k-1}}{2^n} = \frac{1}{2^{k+1}}.$$

Откуда

$$\rho(R_k) = \lim_{n\to\infty} \rho_n(R_k) = \lim_{n\to\infty} \frac{1}{2^{k+1}} = \frac{1}{2^{k+1}} > 0.$$

Кроме того, $R_k \cap R_l = \varnothing$ для всех $k \neq l$. Теперь для любого множества $A \subseteq \omega$ определим

$$\mathcal{R}(A) = \bigcup_{k\in A} R_k.$$

Лемма 4.3. *Для любого множества $A \subseteq \omega$ имеет место $A \leq_{gc} \mathcal{R}(A)$.*

Доказательство. Определим вычислимую функцию $f : \omega^2 \to \omega$ для любой пары (k, m) следующим образом: $f(k, m)$ есть m-е в порядке возрастания число множества R_k. Легко видеть, что функция f осуществляет генерическую клонирующую сводимость множества A к множеству $\mathcal{R}(A)$. □

Также как и в классической теории вычислимости, множество $C \subseteq \omega$ называется *gc-полным* для некоторого класса множеств, если $S \leq_{gc} C$ для любого множества $S \subseteq \omega$ из этого класса.

Теорема 4.2. *Существуют gc-полные множества в классе рекурсивно перечислимых множеств натуральных чисел.*

Доказательство. Пусть $C \subseteq \omega$ – m-полное рекурсивно перечислимое множество. Докажем, что $\mathcal{R}(C)$ будет gc-полным рекурсивно перечислимым множеством. Действительно, для любого рекурсивно перечислимого множества $A \subseteq \omega$ имеет место $A \leq_m C$ с помощью некоторой вычислимой функкции $g : \omega \to \omega$. По лемме 4.3, $C \leq_{gc} \mathcal{R}(C)$ с помощью описанной в доказательстве леммы 4.3 функции $f : \omega^2 \to \omega$. Теперь легко видеть, что $A \leq_{gc} \mathcal{R}(C)$ с помощью функции $h : \omega^2 \to \omega$, которая для любой пары (x, n) определена следующим образом

$$h(x, n) = f(g(x), n).$$

□

4.3 Генерическая тьюрингова сводимость

Пусть A – произвольное множество натуральных чисел. *Генерическим оракулом* множества A называется функция $\varphi_A : \omega \to \{0, 1, ?\}$ такая, что

1. Множество $\{x : \varphi_A(x) =?\}$ пренебрежимо.

2. $\forall x \in \omega\ \varphi_A(x) = 1 \Rightarrow x \in A$.

3. $\forall x \in \omega\ \varphi_A(x) = 0 \Rightarrow x \notin A$.

Множество $A \subseteq \omega$ *генерически сводится по Тьюрингу* к множеству $B \subseteq \omega$, если существует машина M с командами обращения к оракулу такая, что для любого генерического оракула φ_B генерический алгоритм M^{φ_B} вычисляет характеристическую функцию A. Обозначается это $A \leq_{gT} B$.

Будем писать $A \equiv_{gT} B$, если $A \leq_{gT} B$ и $B \leq_{gT} A$. Определим также генерическую тьюрингову степень множества A как

$$d_{gT}(A) = \{B \subseteq \omega : B \equiv_{gT} A\}.$$

Генерическая тьюрингова степень рекурсивно перечислима, если содержит хотя бы одно рекурсивно перечислимое множество. Как принято в классической теории вычислимости, генерические тьюринговы степени будем обозначать маленькими жирными латинскими буквами $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ и т.д. Будем писать, что $\mathbf{a} \leq \mathbf{b}$, если существуют $A \in \mathbf{a}$ и $B \in \mathbf{a}$ такие, что $A \leq_{gT} B$. Аналогично определяется отношение $\mathbf{a} < \mathbf{b}$. Непосредственно из определения генерической тьюринговой сводимости следует, что для любого генерически разрешимого множества $A \subseteq \omega$ и произвольного множества $B \subseteq \omega$ имеет место $A \leq_{gT} B$ – генерический алгоритм для вычисления характеристической функции A просто не использует генерический оракул для B. Поэтому все генерически разрешимые множества образуют одну генерическую тьюрингову степень, которая обозначается $\mathbf{0}$. Кроме того, для любой генерической тьюринговой степени $\mathbf{a}$ имеет место $\mathbf{0} \leq \mathbf{a}$.

Докажем простейшие свойства генерической тьюринговой сводимости.

Предложение 4.9. *Пусть* $A, B, C \subseteq \omega$.

1. $A \leq_{gT} A$.
2. *Если* $A \leq_{gT} B$ *и* $B \leq_{gT} C$, *то* $A \leq_{gT} C$.
3. *Если* $A \leq_{gT} B$ *и* B *генерически разрешимо, то и* A *генерически разрешимо.*

Доказательство. Пункт 1 очевиден.

Докажем пункт 2. Пусть сводимость $A \leq_{gT} B$ реализуется машиной M_1, а сводимость $B \leq_{gT} C$ машиной M_2. Тогда машина M_3, реализующая сводимость $A \leq_{gT} C$ с произвольным генерическим оракулом φ_C работает следующим образом. Сначала реализует с помощью $M_2^{\varphi_C}$ генерический алгоритм для множества B, а затем подает его в машину M_1 в качестве генерического оракула φ_B для множества B. Машина $M_1^{\varphi_B}$ реализует генерический алгоритм для вычисления характеристической функции множества A. Таким образом, $A \leq_{gT} C$.

Докажем пункт 3. Пусть существует генерический алгоритм $\mathcal{A}$, вычисляющий характеристическую функцию множества B. Пусть сводимость $A \leq_{gT} B$ реализуется некоторой машиной M, имеющей команды обращения к генерическому оракулу φ_B множества B. Заменим в машине M команды обращения к оракулу φ_B на вызов генерического алгоритма $\mathcal{A}$. Получим генерический алгоритм, вычисляющий характеристическую функцию множества A. Таким образом, множество A генерически разрешимо. □

Следующее утверждение показывает связь генерической тьюринговой сводимости со сводимостями, введенными в предыдущих подразделах.

Предложение 4.10. *Пусть* $A, B \subseteq \omega$.

1. *Если* $A \leq_{gm} B$, *то* $A \leq_{gT} B$.

2. *Если* $A \leq_{gc} B$, *то* $A \leq_{gT} B$.

Доказательство. Докажем пункт 1. Пусть $A \leq_{gm} B$ с помощью некоторой вычислимой функции $f : \omega \to \omega$. Машина с генерическим оракулом M, реализующая для любого генерического оракула φ_B генерический алгоритм M^{φ_B}, который вычисляет характеристическую функцию A, работает на входе $x \in \omega$, следующим образом. Вычисляется $f(x)$, а затем выдается ответ $\varphi_B(f(x))$. Чтобы убедиться, машина M реализует генерический алгоритм, допустим, что множество

$$S = \{x \in \omega : \varphi_B(f(x)) =?\}$$

непренебрежимо. Но тогда, так как $A \leq_{gm} B$ с помощью f, множество

$$f(S) = \{y \in \omega : \exists x \; y = f(x), \; \varphi_B(y) =?\}$$

тоже непренебрежимо. Но этого не может быть, так как оно является подмножеством пренебрежимого множества

$$\{y \in \omega : \varphi_B(y) =?\}.$$

Пункт 1 доказан.

Докажем пункт 2. Пусть $A \leq_{gc} B$ с помощью некоторой вычислимой функции $f : \omega^2 \to \omega \cup \{?\}$. Машина с генерическим оракулом M, реализующая для любого генерического оракула φ_B генерический алгоритм M^{φ_B}, который вычисляет характеристическую функцию A, работает на входе $x \in \omega$, следующим образом.

1. Вычисляет $f(x,0)$.

2. Если $f(x,0) =?$, то выдает «?» как результат.

3. Если $f(x,0) \neq?$, то присваивает $i := 0$.

4. Обращается к оракулу $\varphi_B(f(x,i))$.

5. Если $\varphi_B(f(x,i)) \neq?$, то выдает $\varphi_B(f(x,i))$ как результат.

6. Если $\varphi_B(f(x,i)) =?$, то увеличивает $i := i+1$ и возвращается на шаг 4.

Этот алгоритм всегда останавливается, так как непренебрежимое множество $\{f(x,n) : n \in \omega\}$ имеет непустое пересечение с генерическим множеством $\{x : \varphi_B(x) \neq?\}$. То, что этот алгоритм является генерическим, следует из определения генерической клонирующей сводимости. Корректность алгоритма также следует из этого определения. Пункт 2 доказан. □

Следующее утверждение показывает, что генерическая тьюрингова сводимость является наиболее общим типом генерической сводимости.

Предложение 4.11. *Имеет место следующее.*

1. *Существуют $A, B \subseteq \omega$ такие, что $A \leq_{gT} B$, но $A \not\leq_{gm} B$.*

2. *Существуют $A, B \subseteq \omega$ такие, что $A \leq_{gT} B$, но $A \not\leq_{gc} B$.*

Доказательство. Докажем пункт 1. При доказательстве утверждения 4.3, было показано, что если A – любое рекурсивное пренебрежимое бесконечное множество и B – любое рекурсивное генерическое множество, не являющееся ко-конечным, то $A \not\leq_{gm} B$. Покажем, что $A \leq_{gT} B$. Машина с генерическим оракулом M, реализующая для любого генерического

оракула φ_B генерический алгоритм M^{φ_B}, который вычисляет характеристическую функцию A, работает на входе $x \in \omega$, следующим образом. Она просто вычисляет характеристическую функцию множества A не обращаясь ни к какому оракулу. Это возможно, так как множество A рекурсивно.

Докажем пункт 2. При доказательстве теоремы 4.8, было показано, что если B – пренебрежимое простое множество, то для любого $A \neq B$ имеет место $A \not\leq_{gc} B$. Если теперь взять в качестве A рекурсивное множество, то, аналогично тому, как это было сделано при доказательстве пункта 1, можно показать, что $A \leq_{gT} B$. □

Следующее утверждение сравнивает генерическую тьюрингову сводимость с классической тьюринговой сводимостью.

Предложение 4.12. *Существуют $A, B \subseteq \omega$ такие, что $A \equiv_T B$, но $A \not\leq_{gT} B$.*

Доказательство. Пусть A – простое пренебрежимое множество. Существование таких множеств было доказано в лемме 2.28. Рассмотрим множество

$$B = A^2 = \{a^2 : a \in A\}.$$

Очевидно, $A \equiv_T B$. Но A супер неразрешимо по лемме 2.27, а B генерически разрешимо. Действительно, генерический алгоритм для B на входе $x \in \omega$ работает следующим образом: проверяет, является ли x квадратом натурального числа, если нет, то выдает ответ 0, иначе «?». Поэтому $A \not\leq_{gT} B$. □

Следующая лемма понадобится в дальнейшем.

Лемма 4.4. *Пусть A – рекурсивно перечислимое множество и $A = A_1 \cup A_2$, где A_1, A_2 – непересекающиеся рекурсивно перечислимые множества. Тогда*

1. *$A_1 \leq_{gT} A$ и $A_2 \leq_{gT} A$.*
2. *Пусть B – произвольное множество. Если $A_1 \leq_{gT} B$ и $A_2 \leq_{gT} B$, то $A \leq_{gT} B$.*

Доказательство. Докажем пункт 1. Машина M с обращениями к любому генерическому оракулу φ_A для A, которая будет сводить A_1 к A работает на входе x следующим образом.

1. Обращается к оракулу $\varphi_A(x)$.
2. Если $\varphi_A(x) =?$, то выдает «?».
3. Если $\varphi_A(x) = 0$, выдает 0.
4. Если $\varphi_A(x) = 1$, то перечисляет множества A_1 и A_2 до тех пор, пока не получит x.
5. Если x получился при перечислении A_1, то выдает 1, иначе выдает 0.

Легко видеть, что этот алгоритм является генерическим.

Докажем пункт 2. Пусть $A_1 \leq_{gT} B$ с помощью машины M_1 и $A_2 \leq_{gT} B$ с помощью машины M_2. Машина M, которая генерически сводит $A_1 \cup A_2$ к B для генерического оракула φ_B работает на x следующим образом.

1. Вычисляет $M_1^{\varphi_B}(x)$ и $M_2^{\varphi_B}(x)$.
2. Если $M_1^{\varphi_B}(x) =?$ или $M_2^{\varphi_B}(x) =?$, выдает «?».
3. Если $M_1^{\varphi_B}(x) = 0$ и $M_2^{\varphi_B}(x) = 0$, выдает 0.

4. Иначе выдает 1.

□

4.4 Генерическая теорема Сакса о разложении

Теорема 4.3. *Пусть A – супер неразрешимое рекурсивно перечислимое множество. Тогда $A = B_0 \cup B_1$, где B_0, B_1 – непересекающиеся рекурсивно перечислимые множества такие, что $A \not\leq_{gT} B_0$ и $A \not\leq_{gT} B_1$.*

Доказательство. Мы будем строить множества B_i, $i = 0, 1$, так, чтобы A было генерически неразрешимым на машинах с оракулами (обычными всюду определенными, а не генерическими) для множеств B_i, $i = 0, 1$. Так как условие $A \leq_{gT} B_i$ означает, что A генерически разрешимо на машине с любым генерическим оракулом для B_i, $i = 0, 1$ (в том числе и всюду определенным), то из этого будет следовать, что $A \not\leq_{gT} B_i$, $i = 0, 1$. Пусть M_e, $e \in \omega$ – эффективная нумерация машин с командами обращения к оракулу, причем все машины могут выдавать только выходы из $\{0, 1, ?\}$. Для построения будем использовать метод приоритета, добиваясь того, чтобы любая машина M_e, $e \in \omega$ с оракулом B_i, $i = 0, 1$, в том случае, если она реализует некоторый генерический алгоритм, ошибалась на некотором входе при вычислении A.

Пусть $\{A_s\}_{s\in\omega}$ – вычислимое перечисление множества A без повторений, то есть такое, что $A_0 = \varnothing$ и $|A_{s+1} \setminus A_s| = 1$ для всех s. Будем строить вычислимые перечисления $\{B_{i,s}\}_{s\in\omega}$, $i = 0, 1$, удовлетворяющие положительному требованию

$$P : x \in A_{s+1} \setminus A_s \Rightarrow x \in B_{0,s+1} \vee x \in B_{1,s+1}$$

и отрицательным требованиям

$$N_{(e,i)} : A \text{ генерически не вычислимо машиной } M_e^{B_i}$$

для любых $e \in \omega$, $i = 0, 1$.

Шаг $s = 0$. Полагаем $B_{i,0} = \varnothing$, $i = 0, 1$.

Шаг $s + 1$. По $B_{i,s}$ определим вычислимую функцию длины

$$l_i(e, s) = \max\{x : (x \leq s) \,\&(\forall y < x \; (M_{e,s}^{B_{i,s}}(y) \downarrow = A_s(y) \vee M_{e,s}^{B_{i,s}}(y) \downarrow =?))\}.$$

Здесь через $M_{e,s}^{B_{i,s}}(y) \downarrow$ обозначается результат работы машины M_e с оракулом $B_{i,s}$ на входе y, если машина остановилась за не более, чем s шагов, и не определено иначе. Через $A_s(y)$ обозначено значение характеристической функции конечного множества A_s. Также определим вычислимую функцию

$$r_i(e, s) = \max\{u(B_{i,s}; e, x, s) : x \leq l_i(e, s)\}.$$

Здесь через $u(B_{i,s}; e, x, s)$ обозначено 1 плюс максимальное число, используемое в вычислении машины M_e при обращении к оракулу $B_{i,s}$ на входе x, если M_e останавливается за не более, чем s шагов и выдает результат, отличный от «?», и 0 иначе.

Пусть $x \in A_{s+1} \setminus A_s$. Выбираем (e', i') как наименьшее в лексикографическом порядке (e, i) такое, что $e \leq s$ и $x \leq r_i(e, s)$, и перечисляем x в $B_{1-i,s+1}$. Тем самым, мы выбираем требование $N_{(e',i')}$ с наивысшим приоритетом, которое нарушается перечислением x в $B_{i'}$, и перечисляем x в другую часть $B_{1-i'}$. Если такого (e', i') не существует, то перечисляем x в B_0. Предельные множества B_i, $i = 0, 1$ в этом процессе и будут искомыми.

Будем говорить, что требование $N_{(e,i)}$ нарушается на шаге $s + 1$ элементом x, если $x \leq r_i(e, s)$ и $x \in B_{i,s+1} \setminus B_{i,s}$. Докажем по индукции, что для любого (e, i) существует $\lim_s r_i(e, s)$ и он конечен. Пусть это так для любого $(k, j) < (e, i)$. Выберем t так, что $r_j(k, s) = r_j(k)$ для всех $(k, j) < (e, i)$ и любого $s > t$. Пусть r больше, чем все такие $r_j(k)$. Су-

ществует такое $v > t$, что первые r членов множества A совпадают с первыми r членами множества A_v. Теперь требование $N_{(e,i)}$ никогда не нарушается после шага v, поэтому существует $\lim_s r_i(e,s)$ и он конечен.

Докажем, что A не является генерически вычислимым на машине $M_e^{B_i}$, $i = 0, 1$, для любого e. Допустим противное, что $M_e^{B_i}$ генерически вычисляет A. Заметим, что найдутся x такие, что $M_e^{B_i}(x) \neq ?$. Тогда $\lim_s l_i(e,s) = \infty$. Выберем s' так, что требование $N_{(e,i)}$ после шага s' никогда не нарушается. Построим теперь генерический алгоритм для множества A, что будет противоречить генерической неразрешимости A. Для вычисления $A(n)$ для любого $n \in \omega$ ищем наименьшее $s > s'$ такое, что $l_i(e,s) > n$.

Докажем индукцией, что для любого $t \geq s$ имеем $l_i(e,t) > n$. Для $t = s$ это очевидно. Пусть это неравенство выполняется для некоторого t, докажем его для $t+1$. Из определения $r_i(e,t)$ и условия $s > s'$ следует, что первые z членов множеств A_{t+1} и A_t совпадают для всех z, использованных в вычислении $M_{e,t}^{B_{i,t}}(x)$ для всех $x \leq n$. Поэтому $M_{e,t+1}^{B_{i,t+1}}(x) = M_{e,t}^{B_{i,t}}(x)$, так что либо $l_i(e,t+1) > n$, либо $A_{t+1} \neq A_t(x)$ для некоторого $x < l_i(e,t)$. Но если $A_t(x) \neq A_s(x)$ для $t \geq s$ и минимального $x \leq n$, то $A_t(x) \neq M_e^{B_{i,t}}(x)$ для любого t, причем $M_e^{B_{i,t}}(x) \neq ?$. То есть $A(x) \neq M_{B_i}(x)$ и $M_e^{B_i}$ генерически не вычисляет A.

Таким образом, для любого $t \geq s$ имеем $l_i(e,t) > n$, откуда следует, что

$$r_i(e,t) \geq \max\{u(B_{i,s}; e, x, s) : x \leq n\}.$$

Поэтому

$$M_{e,s}^{B_{i,s}}(n) = M_e^{B_{i,s}}(n) = M_e^{B_i}(n) = A(n).$$

Так как множество n, на которых данный алгоритм выдает неопределен-

ный ответ «?», совпадает с множеством

$$\{n : M_e^{B_i}(n) =?\},$$

то A генерически разрешимо. Полученное противоречие доказывает теорему. □

4.5 Структура рекурсивно перечислимых степеней генерической тьюринговой сводимости

Рекурсивно перечислимое множество A будем называть *gT-полным*, если для любого рекурсивно перечислимого B имеет место $B \leq_{gT} A$. Соответствующая степень называется *gT-полной*.

Теорема 4.4. *Существуют gT-полные генерические рекурсивно перечислимые степени.*

Доказательство. Существование gm-полных рекурсивно перечислимых множеств было установлено в теореме 4.1. Из предложения 4.10 следует, что любое такое множество является и gT-полным. □

Напомним конструкцию из [55]. Для любого $k \in \omega$ определим следующее множество

$$R_k = \{m \in \omega : 2^k \mid m,\ 2^{k+1} \nmid m\}.$$

Теперь для любого множества $S \subseteq \omega$ определим

$$\mathcal{R}(S) = \bigcup_{k \in S} R_k.$$

Лемма 4.5. *Выполнено следующее:*

1. *Для любого множества $A \subseteq \omega$ имеет место $A \leq_{gT} \mathcal{R}(A)$.*

2. Для любых множеств $A, B \subseteq \omega$ имеет место $A \leq_T B$ тогда и только тогда, когда $\mathcal{R}(A) \leq_{gT} \mathcal{R}(B)$.

Доказательство. Докажем пункт 1. По лемме 4.3 имеет место $A \leq_{gc} \mathcal{R}(A)$. Откуда, по предложению 4.10, $A \leq_{gT} \mathcal{R}(A)$.

Докажем пункт 2. Пусть $A \leq_T B$ с помощью машины M_1 с обычным оракулом B. Тогда машина M_2, реализующая сводимость $\mathcal{R}(A) \leq_{gT} \mathcal{R}(B)$ с произвольным генерическим оракулом $\varphi_{\mathcal{R}(B)}$ работает следующим образом на входе $x \in \omega$. Сначала находит такое k, что $x \in R_k$. Затем запускает машину M_1 на входе k и каждый раз, когда она обращается к оракулу $y \in B?$, находит такое m, что $y \in R_m$ и заменяет это обращение серией обращений

$$\varphi_{\mathcal{R}(B)}(r_i),\ i = 0, 1, 2, \ldots,$$

где $R_m = \{r_0, r_1, \ldots\}$ – перечисление множества R_m в порядке возрастания. Так как множество R_m непренебрежимо, то обязательно найдется j такое, что $\varphi_{\mathcal{R}(B)}(r_j) \neq ?$ и описанная процедура правильно смоделирует обращение к оракулу $y \in B?$. Таким образом, машина $M_2^{\varphi_{\mathcal{R}(B)}}$ реализует всюду определенный (а не генерический) алгоритм для вычисления характеристической функции множества A, а с ним, и множества $\mathcal{R}(A)$. Это означает, что $\mathcal{R}(A) \leq_{gT} \mathcal{R}(B)$.

Обратно, пусть $\mathcal{R}(A) \leq_{gT} \mathcal{R}(B)$ с помощью машины M_1 с любым генерическим оракулом $\varphi_{\mathcal{R}(B)}$. В частности, можно взять обычный оракул для множества $\mathcal{R}(B)$, команды обращения к которому вида $a \in \mathcal{R}(B)$ можно смоделировать следующей процедурой. Сначала ищем такое k, что $a \in R_k$, а затем делаем запрос $k \in B?$. Таким образом, получилась машина M_2 с обычным оракулом B, реализующая генерический алгоритм распознавания множества $\mathcal{R}(A)$. Теперь машина M_3 с оракулом

B, распознающая множество A работает на входе $x \in \omega$ следующим образом. Перебирает элементы $r_1, r_2, \ldots$ множества R_x в порядке возрастания до тех пор пока не получит элемент r_i такой, что $M_2^B(r_i) \neq ?$. Такой элемент обязательно встретится, так как множество R_x непренебрежимо. При этом, очевидно, $x \in A \Leftrightarrow M_2^B(r_i) = 1$. Это означает, что $A \leq_T B$. □

Следующее утверждение является аналогом классической теоремы Мучника-Фридберга о существовании несравнимых относительно тьюринговой сводимости рекурсивно перечислимых степеней.

Теорема 4.5. *Существуют несравнимые генерические тьюринговы рекурсивно перечислимые степени.*

Доказательство. Пусть A и B – рекурсивно перечислимые множества, несравнимые относительно тьюринговой сводимости. Тогда, по лемме 4.5, рекурсивно перечислимые множества $\mathcal{R}(A)$ и $\mathcal{R}(B)$ будут несравнимы относительно генерической тьюринговой сводимости. Несравнимыми будут и их степени. □

Будем называть генерическую тьюрингову рекурсивно перечислимую степень $\mathbf{a}$ *максимальной*, если не существует неполной генерической тьюринговой рекурсивно перечислимой степени $\mathbf{b}$ такой, что $\mathbf{a} < \mathbf{b}$.

Теорема 4.6. *Не существует максимальной генерической рекурсивно перечислимой степени.*

Доказательство. Пусть $\mathbf{a}$ – неполная генерическая тьюрингова рекурсивно перечислимая степень. В ней найдется рекурсивно перечислимое множество A, которое не является gT-полным. По теореме Сакса о плотности полурешетки тьюринговых рекурсивно перечислимых степеней,

существует такое неполное относительно классической тьюринговой сводимости рекурсивно перечислимое множество B, что $A <_T B$. Теперь $A \leq_{gT} \mathcal{R}(A) <_{gT} \mathcal{R}(B)$ по лемме 4.5. Таким образом, для генерической тьюринговой степени $\mathbf{b}$ множества $\mathcal{R}(B)$ будет выполняться $\mathbf{a} < \mathbf{b}$. Это означает, что степень $\mathbf{a}$ не может быть максимальной. □

Ненулевая генерическая тьюрингова рекурсивно перечислимая степень $\mathbf{a}$ называется *минимальной*, если не существует такой генерической тьюринговой рекурсивно перечислимой степени $\mathbf{b}$ что $\mathbf{0} < \mathbf{b} < \mathbf{a}$.

Теорема 4.7. *Не существует минимальной генерической тьюринговой рекурсивно перечислимой степени.*

Доказательство. Пусть $\mathbf{a}$ – ненулевая генерическая тьюрингова рекурсивно перечислимая степень. В ней найдется рекурсивно перечислимое множество A, которое не является генерически разрешимым. По теореме 4.3 $A = A_0 \cup A_1$, где A_0, A_1 – непересекающиеся рекурсивно перечислимые множества такие, что $A_0 <_{gT} A$ и $A_1 <_{gT} A$. По крайней мере одно из них (для определенности, пусть это будет A_0) должно быть генерически неразрешимым, так как иначе, объединение двух генерически разрешимых множеств $A = A_0 \cup A_1$ будет тоже генерически разрешимым. Обозначим через $\mathbf{b}$ генерическую тьюрингову степень множества A_0. Теперь имеет место $\mathbf{0} < \mathbf{b} < \mathbf{a}$. Таким образом, минимальной генерической тьюринговой рекурсивно перечислимой степени не существует. □

Остается открытым вопрос: выполнен ли генерический аналог теоремы Сакса о плотности полурешетки тьюринговых степеней?

5 Генерическая полиномиальная сводимость

5.1 Определение и простейшие свойства

Пусть I, J — некоторые множества входов с определенными на них функциями размера. Множество $A \subseteq I$ *генерически полиномиально сводится* к множеству $B \subseteq J$ (обозначается $A \leq_{GenP} B$), если существуют вероятностный полиномиальный алгоритм $\mathcal{R} : I \times \mathbb{N} \to P(J) \cup \{?, !\}$, полином $p(n)$, полином $q(n)$ степени больше 2 и константа $C > 0$, такие, что

1. для всех $x \in I$ либо $\forall n\, \mathcal{R}(x, n) = \{?\}$, либо для всех $n \geq q(k)$, где $k = size(x)$, выполнены следующие условия:
   (a) $\forall y \in \mathcal{R}(x, n)\ (y \neq !\ \Rightarrow\ size(y) = n)$;
   (b) все элементы в $\mathcal{R}(x, n) \setminus \{!\}$ выдаются алгоритмом $\mathcal{R}$ равновероятно;
   (c) вероятность получить ответ «!» в $\mathcal{R}(x, n)$ не больше 2^{-Ck};
   (d) $\dfrac{|\mathcal{R}(x, n)|}{|J_n|} > \dfrac{1}{(p(n))^k}$;
   (e) $x \in A \Rightarrow \mathcal{R}(x, n) \subseteq B$;
   (f) $x \notin A \Rightarrow \mathcal{R}(x, n) \subseteq J \setminus B$;
2. множество $\{x \in I : \forall n\ (\mathcal{R}(x, n) = ?)\}$ сильно пренебрежимо.

Рассмотрим подробнее каждое из свойств данного определения. Помимо элементов множества J, алгоритм $\mathcal{R}$ может выдавать два особых выхода: «?» — неопределенный ответ и «!» — неудачная генерация. Вероятностный полиномиальный алгоритм $\mathcal{R}$ (если его выход не равен «?») каждому элементу x множества входов I проблемы A и каждому размеру $n \geq q(size(x))$ сопоставляет достаточно большое подмножество (свойство 1d) $\mathcal{R}(x, n)$ входов J проблемы B размера n (свойство 1a). При этом

каждый элемент из $\mathcal{R}(x, n)$, за исключением элемента «!», генерируется равновероятно (свойство 1b). Вероятность получить неудачно сгенерированный выход «!» экспоненциально мала (свойство 1c). Свойства 1e и 1f гарантируют, что это подмножество полностью попадает в B, если $x \in A$, и полностью лежит вне B для $x \notin A$. Свойство 2 говорит о том, что алгоритм $\mathcal{R}$ лишь для пренебрежимого множества входов из I может всегда выдавать неопределенный ответ.

Теорема 5.1. *Если $A \leq_{GenP} B$ и B сильно генерически разрешимо за полиномиальное время, то для A существует полиномиальный вероятностный алгоритм, распознающий A на некотором полиномиальном сильно генерическом множестве. Таким образом, при условии $P = BPP$ множество A сильно генерически разрешимо за полиномиальное время.*

Доказательство. Пусть $\mathcal{R}$ — полиномиальный вероятностный алгоритм, осуществляющий генерическую полиномиальную сводимость $A \leq_{GenP} B$; $\mathcal{B}$ — сильно генерический полиномиальный алгоритм, вычисляющий характеристическую функцию множества B; $q(n)$ — многочлен из определения сводимости $A \leq_{GenP} B$. Вероятностный генерический алгоритм $\mathcal{A}$, вычисляющий характеристическую функцию множества A, работает на входе x размера n следующим образом.

1. Запустить алгоритм $\mathcal{R}(x, q(n))$, который выдает ответ y.
2. Если $y = ?$, выдать «?».
3. Если $y = !$, выдать 0.
4. Иначе запустить алгоритм $\mathcal{B}$ на $y \in J$.
5. Если $\mathcal{B}(y) = ?$, выдать 0.

6. Иначе выдать ответ $\mathcal{B}(y)$.

Очевидно, что этот вероятностный алгоритм полиномиален. Кроме того, неопределенный ответ он выдает на сильно генерическом множестве — это следует из свойства 2 определения генерической полиномиальной сводимости. Заметим, что неправильный ответ алгоритм может выдать на шагах 3 и 5. Докажем, что вероятность этого меньше $1/2$. Для шага 3 вероятность меньше 2^{-Cn} по свойству 1c из определения сводимости — меньше $1/4$ при достаточно большом n. Для шага 5 она не больше

$$\frac{|\{y \in J : \mathcal{B}(y) = ?\}_{q(n)}|}{|\mathcal{R}(x, q(n))|} = \frac{|\{y \in J : \mathcal{B}(y) = ?\}_{q(n)}|}{|J_{q(n)}|} \cdot \frac{|J_{q(n)}|}{|\mathcal{R}(x, q(n))|}.$$

Напомним, что индекс $q(n)$ внизу означает, что мы рассматриваем в соответствующих множествах только элементы размера $q(n)$. Так как алгоритм $\mathcal{B}$ сильно генерический, то существует константа $\alpha > 0$, такая, что

$$\frac{|\{y \in J : \mathcal{B}(y) = ?\}_{q(n)}|}{|J|_{q(n)}} < \frac{1}{2^{\alpha q(n)}}.$$

Из свойства 1d определения сводимости следует, что существует полином $p(n)$, такой, что для любого n

$$\frac{|J|_{q(n)}}{|\mathcal{R}(x, q(n))|} < p(q(n))^n = 2^{n \log(p(q(n)))}.$$

Получаем оценку сверху для вероятности выдачи ответа на шаге 5

$$\frac{2^{n \log(p(q(n)))}}{2^{\alpha q(n)}} = 2^{n \log(p(q(n))) - \alpha q(n)} < 1/4$$

для достаточно больших n, так как многочлен $q(n)$ имеет степень не меньше 2. Итого, получаем, что вероятность неправильного ответа меньше $1/4+1/4 = 1/2$. Таким отбразом, вероятностный алгоритм $\mathcal{A}$ работает корректно. □

Докажем транзитивность полиномиальной генерической сводимости.

Теорема 5.2. *Если* $A \leq_{GenP} B$ *и* $B \leq_{GenP} C$, *то* $A \leq_{GenP} C$.

Доказательство. Пусть $\mathcal{R}_1$, $\mathcal{R}_2$ — полиномиальные вероятностные алгоритмы, осуществляющие генерические полиномиальные сводимости $A \leq_{GenP} B$ и $B \leq_{GenP} C$. Пусть $p_1(n), p_2(n), q_1(n), q_2(n)$ — соответствующие полиномы и C_1, C_2 — соответствующие константы. Вероятностный алгоритм $\mathcal{R}_3$, осуществляющий сводимость $A \leq_{GenP} C$, работает на входе (x, n), где $n > q_2(q_1(k))$, $k = size(x)$, следующим образом:

1. запустить алгоритм $\mathcal{R}_1(x, q_1(k))$, который выдаст ответ y;
2. если $y = ?$, выдать «?»;
3. если $y = !$, выдать «!»;
4. иначе запустить алгоритм $\mathcal{R}_2(y, n)$, который выдаст ответ z;
5. если $z = ?$, выдать «!»;
6. если $z = !$, выдать «!»;
7. иначе выдать z.

Очевидно, что этот вероятностный алгоритм является полиномиальным. Необходимо проверить, что выполнены все свойства полиномиальной генерической сводимости. Свойство 2 выполнено, так как алгоритм выдает неопределенный ответ только на тех элементах x, на которых это делает алгоритм $\mathcal{R}_1$. Свойства 1a, 1b выполняются, так как они выполнены для алгоритма $\mathcal{R}_2$. Свойства 1e, 1f выполняются, так как они верны для алгоритмов $\mathcal{R}_1$ и $\mathcal{R}_2$. Свойство 1d также следует из того, что оно выполнено для алгоритма $\mathcal{R}_2$, при этом соответсвующий полином $p_3(n) = p_2(n)$.

Докажем свойство 1c. Ответ «!» может быть выдан на шагах 3, 5 и 6. Оценим вероятность для каждого шага. Для шага 3 вероятность меньше $2^{-C_1 k}$. На шаге 5 вероятность оценивается как в доказательстве теоремы 5.1: она не больше

$$2^{k \log(p_1(q_1(k))) - \alpha q_1(k)} < 2^{-C_1 k}$$

для достаточно больших k. Для шага 6 эта вероятность меньше $2^{-C_2 size(y)} < 2^{-C_2 k}$ по свойству 1c для сводимости $B \leq_{GenP} C$. Итого получаем оценку сверху

$$2^{-C_1 k} + 2^{-C_1 k} + 2^{-C_2 k} < 2^{-C_3 k}$$

для некоторой константы $C_3 > 0$. □

5.2 Генерически NP-полные проблемы

Определим сильно генерический аналог класса NP. Пусть I – множество входов с определенной на нем функцией размера входа. Множество $S \subseteq I$ принадлежит *классу* sgNP, если существует полиномиальное сильно генерическое множество $G \subseteq I$, такое, что $S \cap G \in$ NP. Множество $S \in$ sgNP называется *генерически NP-полным*, если для любого $A \in$ sgNP имеет место $A \leq_{GenP} S$.

В подразделе 3.2 введено представление булевых формул с помощью бинарных деревьев. Будем использовать это представление чтобы доказать генерическую NP-полноту проблемы выполнимости булевых формул.

Для любой булевой формулы φ определим множество

$$Eq(\varphi) = \{\varphi \vee ((x_1 \wedge \neg x_1) \wedge \psi),\ \psi - \text{произвольная булева формула}\}.$$

Очевидно, что φ выполнима тогда и только тогда, когда любая формула из $Eq(\varphi)$ выполнима.

Лемма 5.1. *Для любой формулы φ имеет место*

$$\frac{|Eq(\varphi) \cap \mathcal{BF}_n|}{|\mathcal{BF}_n|} > \frac{1}{(16(n-2))^k}$$

для любого $n > k+2$, где k — размер формулы φ.

Доказательство. Пусть формула φ имеет размер k. Тогда для любой формулы $\varphi \vee ((x_1 \wedge \neg x_1) \wedge \psi)$ из множества $Eq(\varphi) \cap \mathcal{F}_{n+2}$ формула ψ должна иметь размер $n-k$. Кроме того, в этой формуле может участвовать любая из n переменных. Поэтому можно подсчитать

$$|Eq(\varphi) \cap \mathcal{BF}_{n+2}| = 2^{n-k-1}(2n)^{n-k}C_{n-k-1}.$$

Отсюда

$$\frac{|Eq(\varphi) \cap \mathcal{BF}_{n+2}|}{|\mathcal{BF}_{n+2}|} = \frac{2^{n-k-1}(2n)^{n-k}C_{n-k-1}}{2^{n-1}(2n)^n C_{n-1}} =$$

$$= \frac{1}{(4n)^k} \cdot \frac{C_{n-k-1}}{C_{n-1}} > \frac{1}{(4n)^k} \cdot \frac{1}{4^k} = \frac{1}{(16n)^k}.$$

Здесь была использована лемма 2.20. Таким образом, имеем

$$\frac{|Eq(\varphi)_{n+2}|}{|\mathcal{BF}_{n+2}|} > \frac{1}{(16n)^k},$$

откуда

$$\frac{|Eq(\varphi)_n|}{|\mathcal{BF}_n|} > \frac{1}{(16(n-2))^k}.$$

□

Теорема 5.3. *Проблема выполнимости булевых формул генерически NP-полна.*

Доказательство. Пусть $A \in \mathrm{sgNP}$. Тогда существует полиномиальное сильно генерическое множество G, такое, что $A \cap G \in \mathrm{NP}$. Отсюда, по теореме Кука, следует, что существует классическая полиномиальная сводимость f множества $A \cap G$ к проблеме выполнимости. Полиномиальный вероятностный алгоритм $\mathcal{R}$, генерически сводящий проблему A к проблеме выполнимости, работает на входе (x, n) следующим образом:

1. проверяет, принадлежит ли x множеству G;
2. если $x \notin G$, выдает «?»;
3. если $x \in G$, то строит формулу $\varphi = f(x)$, такую, что φ выполнима тогда и только тогда, когда $x \in A$;
4. случайно и равновероятно генерирует формулу из множества $Eq(\varphi) \cap \mathcal{F}_n$. Как это делается за полиномиальное время, описано в доказательстве теоремы 3.3.

Проверим свойства полиномиальной сводимости. Свойство 2 следует из того, что множество G сильно генерическое. Свойства 1a, 1b следуют из описания шага 4. Свойство 1c очевидно выполняется: алгоритм вообще не выдает ответ «!». Свойства 1e и 1f также выполняются по построению формулы φ. Наконец, свойство 1d следует из леммы 5.1. □

Для доказательства генерической NP-полноты так называемой ограниченной проблемы остановки будем использовать представление машин Тьюринга, введенное в подразделе 2.2. Напомним, что для произвольной машины M через $c(M)$ обозначается множество машин, которые получаются из M добавлением любого количества новых состояний и приписыванием к программе M произвольных правил для новых состояний.

Рассмотрим множество $BHP \subseteq \mathcal{M}$ машин Тьюринга M над алфавитом $\{0,1,\square\}$ таких, что существует $x \in \{0,1\}^*$ такой, что $|x| < size(M)$ и M останавливается на x за $\leq size(M)$ шагов и выдает 1.

Теорема 5.4. *Множество BHP генерически NP-полное.*

Доказательство. Пусть $S \subseteq I$ – любое множество из класса sgNP. Тогда существует полиномиальное сильно генерическое множество $G \subseteq I$ такое, что $S \cap G \in$ NP. Так как $S \cap G \in$ NP, то найдется машина Тьюринга M и полиномы $p(n)$ и $q(n)$ такие, что $x \in S \cap G$ тогда и только тогда, когда найдется $y \in \{0,1\}^*$ такой, что $|y| < p(|x|)$ и M останавливается на (x,y) за $q(|x|+|y|)$ шагов и выдает 1.

Теперь полиномиальный вероятностный алгоритм $\mathcal{R}$, генерически сводящий множество S к BHP, работает на (x,n), где $n > q(|x|+p(|x|))$, следующим образом. Если $x \in G$, то он генерирует случайно и равновероятно машины из множества $c(M^*)_n$, где M^* – это машина из $c(M(x))$ такая, что $size(M^*) > q(|x|+p(|x|))$. Если же $x \notin G$ то выдается множество $\{?\}$.

Проверим, что все условия определения генерической полиномиальной сводимости выполнены. Условия 1a и 1b выполнены очевидным образом. Условие 1c выполнено, так как алгоритм $\mathcal{R}$ вообще не выдает ответ «!». Условие 1d следует из доказательства леммы 2.3. Условия 1e и 1f следуют из того факта, что все машины из множества $c(M(x))$ вычисляют ту же функцию, что и машина $M(x)$. Наконец условие 2 следует из сильной генеричности множества G. Теорема доказана. $\square$

5.3 Генерическая теорема Бейкера-Гилла-Соловея

Определим теперь генерические аналоги классических классов вычислительной сложности P и NP. Пусть I – множество входов с определенной на нем функцией размера входа. Множество $S \subseteq I$ принадлежит *классу* genP, если существует полиномиальный генерический алгоритм, вычисляющий характеристическую функцию множества S. Проблема из класса genP – это проблема, которая решается эффективно (за полиномиальное время), но не для всех входов, а для почти всех входов (входов из генерического множества). Множество $S \subseteq I$ принадлежит *классу* genNP, если существует разрешимое за полиномиальное время генерическое множество $G \subseteq I$ такое, что $S \cap G \in \mathrm{NP}$. Проблема из класса genNP – это проблема, решение которой проверяется эффективно (за полиномиальное время) для почти всех входов (входов из генерического множества G).

Теперь определим релятивизованные версии этих классов. Пусть A есть некоторое подмножество I. *Машина с оракулом* A в своей программе может содержать команды обращения к оракулу, причем выполнение этой команды происходит за один шаг работы вычислительного устройства. Множество $S \subseteq I$ принадлежит *классу* genP^A, если существует полиномиальный генерический алгоритм с оракулом A, вычисляющий характеристическую функцию множества S. Множество $S \subseteq I$ принадлежит *классу* genNP^A, если существует разрешимое за полиномиальное время на машинах с оракулом A генерическое множество $G \subseteq I$ такое, что $S \cap G \in \mathrm{NP}^A$.

Теорема 5.5. *Существуют такие оракулы A и B, что $genP^A = genNP^A$ и $genP^B \neq genNP^B$.*

Доказательство. В качестве оракула A можно взять оракул из теоремы Бейкера-Гилла-Соловея такой, что $\mathrm{P}^A = \mathrm{NP}^A$. Действительно, пусть $S \in \mathrm{genNP}^A$. Тогда существует A-разрешимое за полиномиальное время генерическое множество G такое, что $S \cap G \in \mathrm{NP}^A$. Но $\mathrm{NP}^A = \mathrm{P}^A$, поэтому $S \cap G \in \mathrm{P}^A$. А это означает, что $S \in \mathrm{genP}^A$. Таким образом, $\mathrm{genP}^A = \mathrm{genNP}^A$.

Для построения оракула B будем рассматривать в качестве множества входов I множество $\{0,1\}^*$. Идея построения оракула B такова: построить связанное с ним множество из NP^B так, чтобы любая полиномиальная машина с оракулом B не распознавала это множество на непренебрежимом множестве. Это множество будет следующим:

$$\Omega_B = \{w \in \{0,1\}^* : \text{ в } B \text{ существует строка длины } 2|w|\}$$

Заметим, что $\Omega_B \in \mathrm{NP}^B$ для любого оракула B. Действительно, для входа w сертификатом является строка x длины $2|w|$ такая, что $x \in B$. Если $w \in \Omega_B$, то такой сертификат найдется, иначе нет. Процесс проверки сертификата выполняется за полиномиальное время на машине с оракулом B.

Теперь нужно построить B так, чтобы $\Omega_B \notin \mathrm{genP}^B$. Пусть есть эффективная нумерация полиномиальных машин с оракулами $P_1, P_2, P_3, \ldots$. Образуем из них следующую последовательность

$$\{M_i, i = 1, 2, 3, \ldots\} = \{P_1, P_1, P_2, P_1, P_2, P_3, P_1, P_2, P_3, P_4, P_1, P_2, \ldots\}.$$

В ней каждый раз, после того, как были выписаны машины $P_1, \ldots, P_k$, выписываются машины $P_1, \ldots, P_{k+1}$. Таким образом, каждая полиномиальная машина P_i выписывается бесконечно много раз.

Построение оракула B проходит по шагам, стартуя с множества $B = \{0,1\}^*$, на i-м шаге мы вычеркиваем строки из B так, чтобы машина M_i

не смогла распознать Ω_B на большинстве входов некоторого фиксированного размера. Опишем сначала шаг 1.

Шаг 1. Выберем число k таким, что $p(k) < 2^k$, где $p()$ – это полином, ограничивающий время работы машины M_1. Запускаем M_1 на каждой строке размера k (их 2^k штук). Во время своей работы на каждом входе, как только она делает запрос к оракулу вида $a \in B?$, вычеркиваем эту строку a из B (вынуждая оракул всегда отвечать НЕТ). В процессе всей работы M_1 на всех 2^k входах, мы вычеркнем некоторые $a_1, \ldots, a_m$, где $m < p(k)2^k < 2^{2k}$. Далее смотрим, если M_1 выдает на 2^k входах длины k выход 1 не менее, чем в половине случаев (т.е. $\geq 2^{k-1}$ раз), то вычеркиваем все строки длины $2k$ из B, тем самым гарантируя, что $w \notin \Omega_B$ для любой строки w длины k. Это означает, что машина M_1 ошибается на входах длины k не менее, чем в половине случаев. Если же M_1 выдает в большинстве случаев 0, мы выбираем некоторую строку длины $2k$ из оставшихся в B и помечаем ее (такая строка найдется, так как мы вычеркнули $< 2^{2k}$ строк). Это также гарантирует, что $w \in \Omega_B$ для любой строки w длины k. Таким образом, опять машина M_1 ошибается на входах длины k не менее, чем в половине случаев.

Шаг i. Итак, машины $M_1, \ldots, M_{i-1}$ не распознают Ω_B, ошибаясь на входах с длинами $k_1, \ldots, k_{i-1}$ не менее, чем в половине случаев. Берем $k > k_{i-1}$ и такое, что $p(k) < 2^k$, где $p()$ – это полином, ограничивающий время работы машины M_i. Кроме того, k должно быть больше размеров всех строк, которые были помечены или вычеркнуты на предыдущих шагах. Опять запускаем машину M_i на каждом входе длины k и вычеркиваем строчки-запросы (делая ответы на запросы к оракулу НЕТ), но только те, которые не помечены. Если строка в запросе $a \in B?$ помечена

на предыдущих шагах, она уже точно будет в окончательном оракуле. Это гарантирует то, что работа машины M_i на любом входе w длины k не будет меняться в дальнейшем (то есть ответы на ее запросы $a \in B$ будут теми же и на последующих шагах). Как и на шаге 1, если M_i выдает 1 на большинстве входов длины k, вычеркиваем все строки длины $2k$ из оракула B. Если для большинства выдает 0, то помечаем одну из оставшихся длины $2k$ и переходим к следующему шагу. Опять M_i не может распознать Ω_B для не менее половины входов длины k.

Предельное множество в этом процессе и есть искомый оракул B. Действительно, $\Omega_B \in \mathrm{NP}^B$, а тем более $\Omega_B \in \mathrm{genNP}^B$. Чтобы убедиться, что $\Omega_B \notin \mathrm{genP}^B$ заметим, что для любой полиномиальной машины M с оракулом B множество входов, на которых M дает неправильный ответ, имеет вид

$$E(M) = \bigcup_{i=1}^{\infty} A_i,$$

где

$$A_i = \{w \in \{0,1\}^* : |w| = m_i\}, m_i > m_{i-1},$$

причем $|A_i| \geq 2^{m_i}$ для любого i. Если теперь рассмотреть последовательность

$$\rho_n(E(M)) = \frac{|E(M)_n|}{2^n},\ n = 1, 2, 3, \ldots,$$

то, легко видеть, что $\rho_n(E(M)) \geq \frac{1}{2}$ для бесконечно большого числа значений n. Поэтому множество $E(M)$ непренебрежимо. $\square$

Заключение

В диссертации был развит генерический подход к алгоритмическим проблемам алгебры, математической логики, теории чисел, теории вычислимости и теории вычислительной сложности. Получены следующие основные результаты:

1. Предложен метод генерической амплификации алгоритмических проблем. С его помощью получены следующие результаты:

   (a) Доказана генерическая неразрешимость проблемы остановки для машин Тьюринга.

   (b) Доказана генерическая неразрешимость десятой проблемы Гильберта.

   (c) Доказана генерическая неразрешимость элементарных теорий, неразрешимых в классическом смысле.

   (d) Построен пример полугруппы с генерически неразрешимой проблемой равенства слов.

   (e) Доказано, что арифметика Пресбургера генерически неразрешима за экспоненциальное время.

   (f) Доказано, что проблема выполнимости булевых формул генерически неразрешима за полиномиальное время при условии $\mathrm{P} \neq \mathrm{NP}$ и $\mathrm{P} = \mathrm{BPP}$.

   (g) Доказана генерическая трудноразрешимость проблемы распознавания квадратичных вычетов, проблемы дискретного логарифма, проблемы извлечения корня в группах вычетов, проблемы поиска изоморфизма графов, при условии трудноразрешимости этих проблем в худшем случае.

(h) Доказан генерический аналог теоремы Гёделя о неполноте формальной арифметики.

(i) Доказан генерический аналог теоремы Клини о неподвижной точке.

2. Изучен вопрос о границах применимости метода генерической амплификации: построен пример генерически неразрешимой проблемы, которая не может быть получена с помощью генерической амплификации из неразрешимой в худшем случае проблемы; доказано, что любое рекурсивно перечислимое множество положительной асимптотической плотности может быть получено генерической амплификацией множества натуральных чисел. Доказано существование абсолютно неразрешимых проблем, которые неразрешимы на любом множестве ненулевой асимптотической плотности.

3. Предложены генерические аналоги сводимостей алгоритмических проблем: gm-сводимость, генерическая клонирующая сводимость, генерическая тьюрингова сводимость. Изучена структура рекурсивно перечислимых степеней относительно генерической тьюринговой сводимости: доказано существование полных степеней, доказано существование несравнимых степеней (аналог теоремы Мучника-Фридберга), отсутствие минимальных и максимальных неполных степеней, доказан генерический аналог классической теоремы Сакса о разложении.

4. Предложен генерический аналог полиномиальной сводимости. Доказана генерическая NP-полнота проблемы выполнимости булевых формул и ограниченной проблемы остановки для машин Тьюринга.

5. Доказан генерический аналог теоремы Бейкера-Гилла-Соловея о релятивизации проблемы $\mathrm{P} \neq \mathrm{NP}$.

В диссертации создано новое научное направление, связанное с изучением алгоритмических проблем в алгебре, математической логике, теории чисел, теории вычислимости и теории вычислительной сложности. Полученные результаты содержат решения актуальных задач теории генерической вычислимости и генерической сложности и вносят существенный вклад в развитие теории вычислимости и теории вычислительной сложности.

Результаты и методы диссертации могут быть использованы для дальнейших исследований по теории вычислимости и теории вычислительной сложности. Результаты диссертации также могут быть использованы в образовательном процессе при чтении специальных курсов по теории вычислимости и теории вычислительной сложности, предназначенных для студентов и аспирантов высших учебных заведений.

Список литературы

[1] М. М. Арсланов. О некоторых обобщениях теоремы о неподвижной точке. Изв. вузов. Матем., 5 (1981), 9–16.

[2] И. В. Ашаев. Основы теории алгоритмов, Омск: Издательство ОмГУ, (2006), 172 с.

[3] Л. Д. Беклемишев. Теоремы Гёделя о неполноте и границы их применимости. I. Успехи математических наук, 65:5(395) (2010), 61–106.

[4] Э. Х. Гимади, Н. И. Глебов, В. А. Перепелица. Алгоритмы с оценками для задач дискретной оптимизации. Проблемы кибернетики, 31 (1975), 35–42.

[5] М. Гэри, Д. Джонсон. Вычислительные машины и труднорешаемые задачи. М.: Мир, (1982), 419 с.

[6] Ю. Л. Ершов, И. А. Лавров, А. Д. Тайманов, М. А. Тайцлин. Элементарные теории. УМН, 20:4(124) (1965), 37–108.

[7] Д. Кнут. Искусство программирования. Изд. Вильямс, (2010), 720 с.

[8] Н. Коблиц. Курс теории чисел и криптографии. Москва: ТВП, (2001), 254 с.

[9] А. И. Мальцев. Алгоритмы и рекурсивные функции. М.: Наука, (1965), 394 с.

[10] А. А. Марков. Невозможность некоторых алгорифмов в теории ассоциативных систем. Доклады АН СССР, 55(7) (1947), 587–590.

[11] Ю. В. Матиясевич. Простые примеры неразрешимых канонических исчислений. Труды МИАН СССР, 93 (1967), 50–88.

[12] Ю. В. Матиясевич Диофантовость перечислимых множеств. Доклады Академии наук СССР, 191 (2) (1970), 279–282.

[13] Ю. В. Матиясевич. Десятая проблема Гильберта. Москва: Наука, (1993), 224 с.

[14] Э. Мендельсон. Введение в математическую логику. Москва: Наука, (1976), 320 с.

[15] А. А. Мучник. Неразрешимость проблемы сводимости теории алгоритмов. Доклады АН СССР, 108 (1956), 194– 197.

[16] Х. Роджерс. Теория рекурсивных функций и эффективная вычислимость. М.:Мир, (1972), 624 с.

[17] В. А. Романьков. Введение в криптографию. 2-е изд., исправ., М. : ФОРУМ, (2012). 240 с.

[18] В. А. Романьков. Диофантова криптография на бесконечных группах. Прикладная дискретная математика, 16 (2012), 15–42.

[19] В. А. Романьков. Алгебраическая криптография, Омск: ОмГУ, (2013), 136 с.

[20] Р. И. Соар. Вычислимо перечислимые множества и степени. Казань: Казанское математическое общество, (2000), 576 с.

[21] Н. К. Верещагин, А. Шень. Лекции по математической логике и теории алгоритмов. Часть 3. Вычислимые функции. М.: МЦНМО, (2012), 160 с.

[22] А.М. Вершик, П.В. Спорышев. Оценки среднего числа шагов симплекс-метода и задачи асимптотической интегральной геометрии. Доклады Академии наук СССР, 271 (5) (1983), 1044–1048.

[23] А. М. Вершик, П. В. Спорышев. Асимптотическая оценка среднего числа шагов параметрического симплекс-метода. Журнал вычислительной математики и математической физики, 26 (6) (1986), 813–826.

[24] М. Вялый, А. Китаев, А. Шень. Классические и квантовые вычисления. М.: МЦНМО, ЧеРо, (1999), 192 с.

[25] Г. С. Цейтин. Ассоциативное исчисление с неразрешимой проблемой эквивалентности. Труды МИАН СССР, 52 (1958), 172–189.

[26] U. Andrews, M. Cai, D. Diamondstone, C. Jockusch, S. Lempp. Asymptotic density, computable traceability, and 1-randomness. Fundamenta Mathematicae, 234 (2016), 41–53.

[27] E. Astor. Asymptotic density, immunity, and randomness. Computability, 4 (2) (2015), 141–158.

[28] L. Babai, P. Erdos, S. Selkow. Random graph isomorphism. SIAM Journal of Computing, 9 (3) (1980), 628–635.

[29] T. Baker, J. Gill, R. Solovay. Relativizations of the P=?NP question. SIAM Journal on Computing, 4 (1975), 431–442.

[30] L. Bienvenu, A. Day, R. Holzl. From bi-immunity to absolute undecidability. Journal of Symbolic Logic, 78 (4) (2013), 1218–1228.

[31] L. Bienvenu, D. Desfontaines, A. Shen. Generic algorithms for halting problem and optimal machines revisited. Logical Methods in Computer Science, 12 (2:1) (2016), 1–29.

[32] M. Blum. How to prove a theorem so no one else can claim it. Proceedings of the International Congress of Mathematicians, Berkeley, CA, (1986), pp. 1444–1451.

[33] A. Bogdanov, L. Trevisan. Average-Case Complexity. Electronic Colloquium on Computational Complexity, Report No. 73 (2006).

[34] A. V. Borovik, A. G. Myasnikov, V. N. Remeslennikov. The conjugacy problem in amalgamated products I: regular elements and black holes. International Journal of Algebra and Compututation, 17 (7) (2007), 1299–1333.

[35] A. V. Borovik, A. G. Myasnikov, V. N. Remeslennikov. Generic complexity of the conjugacy problem in HNN-extensions and algorithmic stratification of Miller's groups. Int. J. Algebra Comput., 17 (963) (2007), 963–997.

[36] C.S. Calude, M.A. Stay. Most programs stop quickly or never halt, Advances in Applied Mathematics, 40 (3) (2008), 295–308.

[37] G. J. Chaitin. A theory of program size formally identical to information theory, Journal of ACM, 22 (1975), 329–340.

[38] P. Cholak, G. Igusa. Bounding a density-1 and quasiminimality in the generic degrees. The Journal of Symbolic Logic, 82 (3) 2017, 931–957.

[39] S. Cook. The complexity of theorem proving procedures. Proceedings of the Third Annual ACM Symposium on Theory of Computing, (1971), 151–158.

[40] V. Diekert, A. G. Myasnikov, A. Weiß. Conjugacy in Baumslag's group, generic case complexity, and division in power circuits. Algorithmica, 4 (76) (2016), 961–988.

[41] V. Diekert, A. G. Myasnikov, A. Weiß. Amenability of Schreier graphs and strongly generic algorithms for the conjugacy problem. Journal of Symbolic Computation, 83 (2017), 147–165.

[42] R.G. Downey, C.G. Jockusch, P.E. Schupp. Asymptotic density and computably enumerable sets. Journal of Mathematical Logic, 13 (2013), 43 pp.

[43] R.G. Downey, C.G. Jockusch, T.H. McNicholl, P.E. Schupp, Asymptotic density and the Ershov Hierarchy, Math. Log. Quarterly, 61 (2015), 189–195.

[44] R. Gilman, A. G. Miasnikov, A. D. Myasnikov, A. Ushakov. Report on generic case complexity. Herald of Omsk University, Special Issue, (2007), 103–110.

[45] R. Gilman, A. G. Miasnikov, D. Osin. Exponentially generic subsets of groups. Illinois J. Math., 1 (54) (2010), 371–388.

[46] R. Gilman, A. G. Miasnikov, V. A. Roman'kov. Random equations in nilpotent groups. Journal of Algebra, 1 (352) (2012), 192–214.

[47] Y. Gurevich. Average case completeness. Journal of Computer and System Sciences, 42 (1991), 346–398.

[48] Y. Gurevich, S. Shelah. Expected computation time for Hamiltonian path problem. SIAM Journal on Computing, 3 (16) (1987), 486–502.

[49] M.J. Fischer, M.O. Rabin. Super-Exponential Complexity of Presburger Arithmetic. Proceedings of the SIAM-AMS Symposium in Applied Mathematics, 7 (1974), 27–41.

[50] R. M. Friedberg. Two recursively enumerable sets of incomparable degrees of unsolvability. Prod. Nat. Acad. Sci. USA, 43 (1957), 236–238.

[51] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. I. Monatshefte für Mathematik und Physik, 38 (1931), 173–198.

[52] J. D. Hamkins and A. Miasnikov. The halting problem is decidable on a set of asymptotic probability one. Notre Dame Journal of Formal Logic, 47 (4) (2006), 515–524.

[53] D. Hirschfeldt, C. Jockusch, R. Kuyper, P. Schupp. Coarse reducibility and algorithmic randomness. Journal of Symbolic Logic, 81 (3) (2016), 1028–1046.

[54] D.R. Hirschfeldt, C.G. Jockusch, T. McNicholl, P.E. Schupp. Asymptotic density and the coarse computability bound. Computability, 5 (2016), 13–27.

[55] C. Jockusch, P. Schupp. Generic computability, Turing degrees, and asymptotic density. Journal of the London Mathematical Society, 85 (2) (2012), 472–490.

[56] J. Jones. Undecidable Diophantine equations. Bull. Amer. Math. Soc., 3 (2) (1980), 859–862.

[57] G. Igusa. Nonexistence of minimal pairs for generic computability. Journal Symbolic Logic, Vol. 78 (2013), No. 2, pp. 511–522.

[58] G. Igusa. The generic degrees of density-1 sets, and a characterization of the hyperarithmetic reals. Journal of Symbolic Logic, 80 (4) (2015), 1290–1314.

[59] Impagliazzo R., Wigderson A. P=BPP unless E has Subexponential Circuits: Derandomizing the XOR Lemma. Proceedings of the 29th STOC, (1997), 220–229.

[60] M. Kambites. Generic Complexity of Finitely Presented Monoids and Semigroups. Computational complexity, 20 (1) (2011), 21–50.

[61] I. Kapovich, A. Myasnikov, P. Schupp, V. Shpilrain. Generic-case complexity and decision problems in group theory. Journal of Algebra, 264 (2003), 665–694.

[62] I. Kapovich, P. Schupp. Genericity, the Arzhantseva-Ol'shanskii method and the isomorphism problem for one-relator groups Mathematische Annalen, 331 (2005), 1–19.

[63] R. Karp. The fast approximate solution of hard combinatorial problems. Graph Theory and Computing, (1975), 15–31.

[64] B. Khoussainov, A. Miasnikov. Finitely presented expansions of groups, semigroups, and algebras. Trans. Amer. Math. Soc., 366 (2014), 1455–1474.

[65] V. Klee, G. Minty. How good is the simplex algorithm? Inequalities, III (Proc. Third Sympos., Univ. California, Los Angeles, Calif., 1969; dedicated to the memory of Theodore S. Motzkin), Academic Press, New York, (1972), 159–175.

[66] S.C. Kleene. On Notation for Ordinal Numbers. Journal of Symbolic Logic, 3 (1938), 150–155.

[67] Levin L. Average case complete problems. SIAM Journal on Computing, 15 (1987), 285–286.

[68] Matiyasevich Yu., Robinson J. Reduction of an arbitrary diophantine equation to one in 13 unknowns. Acta Arithmetica, 27 (1975), 521–553.

[69] A. Myasnikov, D. Osin. Algorithmically finite groups. Journal of Pure and Applied Algebra, 215 (2011), 2789–2796.

[70] A. Myasnikov, V. Romankov. Diophantine cryptography in free metabelian groups: Theoretical base. Groups, Complexity, Cryptology, 6 (2) (2014), 103–120.

[71] A. Miasnikov, P. Schupp. Computational complexity and the conjugacy problem. Computability, 4 (6) (2017) 307–318.

[72] A. Myasnikov, V. Shpilrain, A. Ushakov. Non-commutative cryptography and complexity of group-theoretic problems. Mathematical Surveys and Monographs, American Mathematical Society, 177 (2011).

[73] A. Myasnikov, A. Ushakov. Random van Kampen Diagrams and algorithmic problems in groups. Groups Complexity Cryptology, 3 (1) (2011), 121–185.

[74] B. Poonen, J. F. Voloch. Random Diophantine Equations. Arithmetic of higher dimensional algebraic varieties, Progress in Mathematics 226, Birkhäuser (2004), 175–184.

[75] E. L. Post. Recursive unsolvability of a problem of Thue. Journal of Symbolic Logic, 1 (12) (1947), 1–11.

[76] R. Rivest, A. Shamir, L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. Commun. ACM, 21 (2) (1978), 120–126.

[77] G. E. Sacks. On the degrees less than $0'$. Annals of mathematics, 77 (2) (1963), 211–231.

[78] G. E. Sacks. The recursively enumerable degrees are dense. Annals of mathematics, 80 (2) (1964), 300–312.

[79] T. Skolem. Diophantische Gleichungen. Berlin: Springer, (1938).

[80] S. Smale. On the average number of steps of the simplex method of linear programming. Mathematical programming, 27 (3) (1983), 241–262.

[81] J. Wang. Average-case intractable NP problems. Advances in Languages, Algorithms, and Complexity, Kluwer Academic Publishers, (1997), 313–378.

[82] W. Woess. Cogrowth of groups and simple random walks. Archive of Mathematics, 41 (1983), 363–370.

Работы автора по теме диссертации, опубликованные в журналах из списка ВАК

[83] A.N.Rybalov. On the strongly generic undecidability of the Halting Problem. Theoretical Computer Science, 377 (2007), 268–270.

[84] A.G.Myasnikov, A.N.Rybalov. Generic complexity of undecidable problems. Journal of Symbolic Logic, 73 (2) (2008), 656–673.

[85] A.N. Rybalov. Generic Complexity of Presburger Arithmetic. Theory of Computing Systems, 46 (1) (2010), 2–8.

[86] А.Н. Рыбалов. Генерическая сложность теорий первого порядка. Сибирские электронные математические известия, 8 (2011), 168–178.

[87] А.Н. Рыбалов. О генерической неразрешимости Десятой проблемы Гильберта. Вестник Омского университета, 4 (2011), 19–22.

[88] A.N. Rybalov. Generic complexity of the Diophantine problem. Groups Complexity Cryptology, 5 (1) (2013), 25–30.

[89] А.Н. Рыбалов. Генерическая неполнота формальной арифметики. Сибирские электронные математические известия, 12 (2015), 185–189.

[90] А.Н. Рыбалов. О генерической сложности проблемы распознавания квадратичных вычетов. Прикладная дискретная математика, 28 (2015), 54–58.

[91] A.N. Rybalov. On the generic complexity of the searching graph isomorphism problem. Groups Complexity Cryptology, 7 (2) (2015), 191–194.

[92] А.Н. Рыбалов. О генерической сложности элементарных теорий. Вестник Омского университета, 4 (2015), 14–17.

[93] А.Н. Рыбалов. О генерической сложности проблемы дискретного логарифма. Прикладная дискретная математика, 33 (2016), 93–97.

[94] А.Н. Рыбалов. Об одном генерическом отношении рекурсивно перечислимых множеств. Алгебра и логика, 55 (5) (2016), 587–596.

[95] A.N. Rybalov. On the Generic undecidability of the Halting Problem for normalized Turing machines. Theory of Computing Systems, 60 (4) (2017), 671–676.

[96] А.Н. Рыбалов. О генерической NP-полноте проблемы выполнимости булевых формул. Прикладная дискретная математика, 36 (2017), 106–112.

[97] А.Н. Рыбалов. Генерическая теорема Гёделя о неполноте. Алгебра и логика, 56 (3) (2017), 348–353.

[98] А.Н. Рыбалов. Генерическая теорема Клини о неподвижной точке. Сибирские электронные математические известия, 14 (2017), 732–736.

[99] А.Н. Рыбалов. О генерической сложности проблемы разрешимости систем диофантовых уравнений в форме Сколема. Прикладная дискретная математика, 37 (2017), 100–106.

[100] A.N. Rybalov. Generic hardness of the Boolean satisfiability problem. Groups Complexity Cryptology, 9(2) (2017), 151–154.

[101] А.Н. Рыбалов. О генерической NP-полноте ограниченной проблемы остановки. Вестник Омского университета, 4 (2017), 22–25.

[102] А.Н. Рыбалов. О генерической сложности проблемы извлечения корня в группах вычетов. Прикладная дискретная математика, 38 (2017), 95–100.

[103] А.Н. Рыбалов. Релятивизованные генерические классы P и NP. Прикладная дискретная математика, 40 (2018), 100–104.

[104] А.Н. Рыбалов. О структуре рекурсивно перечислимых степеней генерической сводимости. Вестник Омского университета, 2 (2018), 35–41.

[105] A. N. Rybalov. A generic m-reducibility. Lecture Notes in Computer Science, 10936 (2018), 232–237.