ФГБОУ ВПО «Московский государственный
университет имени М. В. Ломоносова»
Механико-математический факультет

На правах рукописи

Мусатов Даниил Владимирович

Комбинаторные методы в теории колмогоровской сложности с ограничением на ресурсы

01.01.06 — математическая логика, алгебра и теория чисел

ДИССЕРТАЦИЯ
на соискание ученой степени
кандидата физико-математических наук

Научный руководитель:
доктор физико-математических наук,
профессор Николай Константинович Верещагин

Москва — 2014

Оглавление

Глава 1

Введение

1.1 Актуальность темы и цели работы

Понятие колмогоровской сложности появилось в 1960-х годах в работах Колмогорова [6], Соломонова [45] и Чейтина [14; 15]. Колмогоровскую сложность можно определять для любых конечных объектов, но достаточно рассматривать двоичные слова, т.е. конечные последовательности из нулей и единиц. Неформально говоря, сложность слова есть сложность его алгоритмического описания, т.е. длина кратчайшей программы, которая печатает это слово на пустом входе. Также рассматривают условную сложность одного слова относительно другого, т.е. длину кратчайшей программы, печатающей первое слово после получения на вход второго. Разумеется, эти определения зависят от того, какой язык программирования используется для описания, но согласно теореме Колмогорова–Соломонова существует оптимальный язык программирования, при котором сложности всех слов минимальны с точностью до аддитивной константы. Это позволяет не ссылаться на конкретный язык программирования и при этом формулировать различные утверждения о связи сложностей различных слов. Эти утверждения формулируются с точностью до аддитивной константы, зависящей только от выбора оптимального языка программирования и не зависящей от конкретных слов. В формулировках эта константа традиционно обозначается через $O(1)$. (Некоторые утверждения формулируются с точностью до другого аддитивного слагаемого: $O(\log n)$, $O(\log^3 n)$ и т.п.) К сожалению, до сих пор не сложилось единого обозначения для сложности. В диссертации сложность слова x обозначается через $\mathrm{C}(x)$, а условная сложность слова x относительно y — через $\mathrm{C}(x|y)$.

Также рассматривают колмогоровскую сложность с ограничением на ре-

сурсы [28]. В этом случае программа, генерирующая слово, должна быть не только короткой, но и использующей ограниченные вычислительные ресурсы, а именно время работы и память. Разумеется, при наложении дополнительных ограничений сложность не уменьшается. Важно, что в отличие от сложности без ограничений сложность с ограничениями является вычислимой функцией.

Для доказательства того, что сложность некоторого слова мала, часто используют следующий принцип, восходящий к Сипсеру [43]: элементы любого небольшого перечислимого множества просты. Более точно: если перечислимое множество S содержит не больше K слов длины n, то все эти слова имеют сложность не больше $\log K + \log n + O(1)$. Действительно, каждое из них можно задать перечисляющим алгоритмом, длиной n и номером в перечислении слов данной длины. Это наблюдение позволяет доказывать различные утверждения о колмогоровской сложности через комбинаторные конструкции: сначала строится некоторое перечислимое множество, затем комбинаторными методами доказывается верхняя оценка на его размер, и, наконец, делается вывод о малой сложности любого входящего в него слова. Более того, если перечисляющий алгоритм вычислительно эффективен, то сложность с ограничением на ресурсы также мала. В диссертации изучаются два примера такого подхода.

Во-первых, изучается теорема Мучника об условном кодировании [30]. Она гласит, что для любых слов a и b, таких что $\mathrm{C}(a|b) < k$, найдётся слово p длины k, для которого одновременно верны соотношения $\mathrm{C}(a|b,p) = O(\log n)$ и $\mathrm{C}(p|a) = O(\log n)$, где $n = \mathrm{C}(a)$. Иными словами, среди всех описаний a при известном b найдётся p, простое относительно a. Принцип доказательства такой: рассматривается некоторое семейство хеш-функций, отображающих слова длины n в слова длины k. Слово p строится как образ слова a, поэтому для его описания требуется лишь задать номер хеш-функции. Для того, чтобы a можно было восстановить по b и p, нужно чтобы p могло быть выбрано как хеш-значение у не слишком большого числа слов, имеющих сложность не больше k при условии b. Этого можно добиться, наложив специальные комбинаторные требования на семейство функций. Сам Мучник использовал свойство экспандера, А.Шень [42] использовал свойство существования онлайн-паросочетаний, а в настоящей работе показано, что можно рассмотреть свойство экстрактора.

Понятие экстрактора было введено в начале 1990-х годов в работе Ни-

сана и Цукермана [35] как технический инструмент. Неформально говоря, экстрактор — это функция, которая получает на вход две «не очень хорошие» случайные величины и превращает их в «почти случайные». В последующие годы было разработано множество конструкций и приложений экстракторов [40; 41; 50], к которым настоящая работа добавляет ещё одно.

Мучник также доказал вариацию теоремы для двух условий. Она гласит, что для любых слов a, b и c, таких что $\mathrm{C}(a|b) < k$ и $\mathrm{C}(a|c) < l$, где $l \leqslant k$, найдётся слово p длины k, такое что для него верны соотношения $\mathrm{C}(a|b,p) = O(\log n)$ и $\mathrm{C}(p|a) = O(\log n)$, а для его начала q длины l верно $\mathrm{C}(a|c,q) = O(\log n)$. Можно обобщить эту теорему на любое константное и даже полиномиальное число условий. Иными словами, программы, превращающие разные слова в a, не только просты относительно a, но и являются префиксами одной и той же длинной программы с точностью до небольших добавок логарифмической длины. Принцип доказательства этой теоремы такой же, но комбинаторное условие на семейство функций будет несколько другим. В настоящей работе вводится понятие префиксного экстрактора и показывается, как при помощи него доказать теорему для двух условий.

Вторым примером параллелизма между сложностными и комбинаторными утверждениями являются теоремы о существовании колмогоровских экстракторов (обычных и усиленных). Колмогоровским экстрактором называется функция двух аргументов, значение которой имеет меньший *дефект случайности*, т.е. разность между длиной и сложностью, чем каждый из её аргументов. Если «условный дефект случайности», т.е. разность между длиной и условной сложностью значения функции при условии одного из аргументов, также уменьшается, то такая функция называется усиленным колмогоровским экстрактором. Колмогоровские экстракторы были определены Дж. Хичкоком и соавторами [18; 22] и затем изучались М. Зимандом [53–55]. Зиманд определил комбинаторные понятия пёстрых и равномерно пёстрых таблиц, доказал их существование и показал, что они являются колмогоровскими экстракторами (обычными и усиленными).

Основной темой диссертации является распространение известных результатов о колмогоровской сложности на сложность с ограничением на ресурсы. Достижений в этом направлении не так много. Можно отметить результат Ли и Ромащенко о симметрии информации [27], а также работу Бюрмана, Ли и ван Мелкебеека о сжатии языков [13]. Для сложности с

ограничением на ресурсы становится важна модель вычислений: использование недетерминизма и/или случайности может существенно уменьшить сложность. В частности, изучают CAM-сложность для недетерминированных вероятностных вычислений. Эта модель была введена Ласло Бабаем [11] и получила метафорическое название игр Артура–Мерлина. Метафора объясняется так: вычисления проводит Артур, беседующий с магом Мерлином. Артур может бросать монетку (т.е. получать случайные биты) и совершать полиномиальные вычисления. Мерлин может вычислять любые функции (даже не вычислимые алгоритмически) и мгновенно узнаёт случайные биты Артура. Взаимодействие устроено так: сначала Артур кидает монетку необходимое число раз, затем Мерлин узнаёт результаты и посылает некоторое сообщение, затем Артур проводит полиномиальные вычисления и выдаёт ответ. Ответом может быть либо двоичное слово, либо символ ошибки $\perp$. Слово x называется результатом работы программы, если с вероятностью хотя бы $\frac{2}{3}$ одновременно выполнены два условия: во-первых, Артур выдаст x при некоторых сообщениях Мерлина; во-вторых, при любых других сообщениях Мерлина Артур выдаст либо то же x, либо $\perp$.

Важным техническим инструментом, используемым в диссертации, является генератор псевдо-случайных чисел Нисана–Вигдерсона [32; 34]. Этот генератор «обманывает» все схемы из функциональных элементов полиномиального размера и константной глубины, т.е. такие схемы выдают асимптотически одинаковый результат на случайном слове и на случайном значении генератора. В диссертации генератор используется «наивным» образом, сродни работе Ромащенко [39]: вместо случайного комбинаторного объекта рассматривается псевдослучайный, что существенно уменьшает область перебора и позволяет доказать теоремы для ограниченных ресурсов.

Итак, основной задачей работы является изучение связей между комбинаторными конструкциями и утверждениями о колмогоровской сложности и использование этих связей для распространения известных теорем о колмогоровской сложности на сложность с ограничением на вычислительные ресурсы.

1.2 Основные результаты

Работа содержит четыре основных результата.

Во-первых, установлена связь теоремы Мучника об условном кодировании и теории экстракторов. С использованием теоремы из [12] в разделе 3.2 доказано, что комбинаторное свойство экстрактора позволяет доказать теорему Мучника об условном кодировании. Также в разделе 3.3 показано, что аналогичной техникой можно получить теорему Мучника для нескольких условий (вплоть до полиномиального числа).

Во-вторых, доказаны аналоги теоремы Мучника для сложности с ограничением на память. Применяются две техники: использование явных экстракторов (теорема 4.1) и «наивная дерандомизация» с использованием генераторов псевдослучайных чисел (теорема 4.2). Комбинацией двух подходов получен аналог теоремы для логарифмической точности при любом ограничении на память (теорема 4.10). Также доказаны аналоги теоремы Мучника для нескольких источников (теорема 4.16 и теорема 4.17). Применение «наивной дерандомизации» позволило доказать теорему не только для полиномиального, но и для экспоненциального числа условий при полиномиальном ограничении на память (теорема 4.18).

В-третьих, доказан аналог теоремы Мучника для CAM-сложности с ограничением на время (теорема 5.1). При доказательстве использовалась техника, разработанная для доказательства CAM-теоремы о сжатии языков [13]. Здесь кодирование осуществляется обычным полиномиальным алгоритмом, а декодирование происходит при помощи алгоритма из класса AM. Также сформулирована гипотеза, обобщающая на CAM-сложность теорему для двух условий (гипотеза 5.8), и проанализированы трудности с обобщением конструкции на этот случай.

В-четвёртых, определены и построены обычные и усиленные колмогоровские экстракторы с ограничением на память (теорема 6.2). Конструкции Зиманда, доказывающие существование таких экстракторов с оптимальными параметрами, упрощены и переложены для новых определений. Использована разработанная при доказательстве аналогов теоремы Мучника техника «наивной дерандомизации».

1.3 Апробация работы

Основные результаты, полученные в работе, были изложены на международных конференциях “Computer Science in Russia” в Новосибирске в 2009 году, в Санкт-Петербурге в 2011 году и в Нижнем Новгороде в 2012 году, опубликованы в сборниках трудов этих конференций’ [60; 62; 64] (серия Lecture Notes in Computer Science, входит в систему Scopus) и специальных выпусках журнала Theory of Computing Systems [59; 61; 63]. В совместных статьях [64] и [63] автору принадлежат метод доказательства теоремы Мучника при помощи экстракторов и теорема для CAM-сложности.

Кроме того, результаты были доложены на следующих научных конференциях и семинарах:

- 8-ая международная конференция по вычислимости, сложности и случайности (CCR), Москва, 2013
- 56-я научная конференция МФТИ, секция дискретной математики, Долгопрудный, 2013
- 55-я научная конференция МФТИ, секция дискретной математики, Долгопрудный, 2012 [65]
- Колмогоровский семинар по сложности вычислений и сложности определений, механико-математический факультет МГУ им. М.В.Ломоносова
- Кафедральный семинар кафедры дискретной математики факультета инноваций и высоких технологий МФТИ(ГУ)
- Семинар по дискретной математике Петербургского отделения математического института им. В.А.Стеклова РАН
- Семинар Давида Гамарника в Массачусетском Технологическом институте (США)

1.4 План дальнейшего изложения

Дальнейший текст организован следующим образом. В главе 2 даны строгие определения всех используемых объектов и дан обзор результатов, которые используются в работе. Также приведены используемые следствия

из известных теорем и доказательства этих следствий. Последующие главы организованы согласно полученным результатам.

В главе 3 излагается доказательство теоремы Мучника об условном кодировании при помощи экстракторов. Сначала даётся формулировка теоремы, делаются несложные замечания по ней и излагается идея доказательства. Затем формулируется свойство, следующее из свойства экстрактора и влекущее теорему Мучника, и доказываются обе импликации. Наконец, метод распространяется на теорему Мучника для нескольких условий, для чего вводится понятие префиксного экстрактора.

В главе 4 формулируются и доказываются различные аналоги теоремы Мучника для сложности с ограничением на память. Первый аналог опирается на существование явных экстракторов. Для существующих явных конструкций и субэкспоненциальных ограничений на память этот метод даёт теорему с точностью $O(\log^3 n)$ вместо $O(\log n)$. Второй аналог доказывает теорему для субэкспоненциальных ограничений с исходной точностью. Для него используется техника «наивной дерандомизации» при помощи генераторов псевдослучайных чисел Нисана–Вигдерсона, которая подробно описывается. Далее два подхода комбинируются для получения теоремы с логарифмической точностью для любого ограничения. Наконец, оба подхода применяются для доказательства теоремы для сложности с ограничением на память для нескольких условий.

В главе 5 техника из статьи [13] применяется для доказательства варианта теоремы Мучника для CAM-сложности. Идея состоит в том, чтобы вместо произвольного явного экстрактора взять экстрактор Тревисана [49], который позволяет быстро осуществлять как кодирование, так и декодирование. Мы формулируем теорему, подробно излагаем конструкцию Тревисана и доказываем, что она корректно работает. Далее формулируется гипотеза об аналогичном утверждении для нескольких условий и анализируются препятствия к расширению конструкции на этот случай.

Наконец, в главе 6 техника «наивной дерандомизации» применяется к теории колмогоровских экстракторов, а именно доказывается существование оптимальных обычных и усиленных колмогоровских экстракторов с ограничением на память. Вначале даются подробные определения и формулировки теорем. Затем, следуя Зиманду, мы вводим понятия «пёстрых» и «равномерно пёстрых» таблиц (наши определения другие, хотя и эквивалентные) и доказываем существование этих объектов с определёнными па-

раметрами. Далее определения слегка модифицируются для применения техники «наивной дерандомизации». Наконец, подробно доказывается, почему (модифицированные) пёстрые таблицы являются колмогоровскими экстракторами с ограничением на память, а (модифицированные) равномерно пёстрые таблицы — усиленными колмогоровскими экстракторами с ограничением на память.

Работа заканчивается небольшим заключением, которое обрисовывает контуры дальнейших исследований, и списком литературы, содержащим 65 наименований, в том числе 7 работ автора по теме диссертации.

1.5 Используемые обозначения

В литературе по колмогоровской сложности нет единой системы обозначений, поэтому мы приведём список используемых обозначений явно:

- Все логарифмы по умолчанию берутся по основанию 2.
- Двоичные слова мы будем обозначать маленькими латинскими буквами, обычно a, b, c, p, x, y, z. Случайное слово будем обозначать буквой r. Множество двоичных слов длины l будем обозначать через $\{0,1\}^l$, множество двоичных слов длины меньше l — через $\{0,1\}^{<l}$, а множество всех двоичных слов — через $\{0,1\}^*$. Пустое слово будем обозначать через ε. Длину слова x будем обозначать через $|x|$.
- Конечные множества будем обозначать большими латинскими буквами, например P, Q, R, S. Количество элементов в множестве P будем обозначать через $|P|$. Кортежи будем обозначать полужирным шрифтом, например $\mathbf{Q}$. Для систем множеств будем использовать каллиграфические буквы, такие как $\mathcal{Q}$, $\mathcal{R}$, $\mathcal{S}$.
- Случайные величины будем обозначать греческими буквами ξ, η и т.п.
- Все распределения вероятностей по умолчанию считаются равномерными на двоичных словах соответствующей длины. Через U_l мы будем обозначать равномерно распределённую случайную величину на словах длины l. Под записью $\Pr_{r\in\{0,1\}^n}[A]$ будем понимать вероятность события A в вероятностном пространстве, где r распределено равномерно на $\{0,1\}^n$. Если длина r ясна из контекста, то указание на неё может быть опущено: $\Pr_r[A]$.

- $\mathrm{C}(x|y)$ обозначает условную колмогоровскую сложность слова x при известном y.
- $\mathrm{C}(x)$ обозначает безусловную колмогоровскую сложность слова x, т.е. $\mathrm{C}(x) = \mathrm{C}(x|\varepsilon)$.
- $\mathrm{C}(x, y)$ обозначает безусловную колмогоровскую сложность пары (x, y).
- $\mathrm{dep}(x, y)$ обозначает зависимость (взаимную информацию) между словами x и y, т.е. величину $\mathrm{C}(x) + \mathrm{C}(y) - \mathrm{C}(x, y)$.
- $\mathrm{C}^{s,t}(x)$ обозначает колмогоровскую сложность слова x для ограничений s на используемую память и t на время работы программы. Если, исходя из контекста, важно только одно из этих ограничений, то второе может опускаться. Условная сложность и сложность пары с ограничениями на ресурсы вводятся аналогично.
- Мы используем обозначения $\mathrm{CN}^{s,t}$, $\mathrm{CBP}^{s,t}$, $\mathrm{CAM}^{s,t}$ для сложности с ограничением на ресурсы для недетермнированных вычислений, вероятностных вычислений и вычислений в модели Артура–Мерлина соответственно. Обычно мы будем опускать индекс, соответствующий ограничению на память.
- Мы используем стандартные асимптотические обозначения:

$$f(n) = O(g(n)), \text{ если } \exists C\, \exists N\, \forall n > N f(n) < C g(n);$$

$$f(n) = \Omega(g(n)), \text{ если } \exists c > 0\, \exists N\, \forall n > N f(n) > c g(n);$$

$$f(n) = \mathrm{poly}(g(n)), \text{ если } \exists c, d\, \exists N\, \forall n > N f(n) < c(g(n) + n + 1)^d.$$

  При этом если выражение с O-, Ω- или poly-обозначением встречается внутри утверждения с кванторами, то цепочки кванторов из приведённых определений должны быть вынесены в самое начало, таким образом константы C, c, d не зависят от всех остальных параметров. В отдельных случаях мы будем прямо указывать, от чего зависят эти константы. (Как правило, от способа описания, использованного при определении колмогоровской сложности).
- Через C_n^k мы обозначаем число сочетаний из n по k, т.е. количество различных k-элементных подмножеств n-элементного множества.

Благодарности

Прежде всего, автор выражает глубокую благодарность своему научному руководителю Николаю Константиновичу Верещагину за постановку задачи о колмогоровских экстракторах, постоянное внимание и поддержку в работе. Автор также благодарит своих соавторов и наставников Андрея Ромащенко и Александра Шеня за знакомство с тематикой, постановку задачи о применении экстракторов к задаче условного кодирования, многочисленные обсуждения, советы и конструктивную критику по тематике диссертации. Автор выражает отдельную благодарность Андрею Альбертовичу Мучнику (1958–2007) за обсуждение результатов автора, развивающих идеи Андрея Альбертовича. Безвременная кончина Андрея Альбертовича стала тяжёлой утратой для российской науки.

Автор благодарит своих коллег Виктора Булатова, Эдуарда Гирша, Брюно Дюрана (Bruno Durand), Илью Ирхина, Дмитрия Ицыксона, Руслана Ишкуватова, Юрия Притыкина, Михаила Раскина и Андрея Румянцева за обсуждение отдельных разделов работы. Автор благодарит Ронена Шалтиела (Ronen Shaltiel) за полезное замечание, позволившее существенно упростить доказательство одного из утверждений. Также автор благодарит участников семинаров кафедры математической логики и теории алгоритмов механико-математического факультета МГУ и кафедры дискретной математики факультета инноваций и высоких технологий МФТИ за внимание и интерес к докладам автора.

Автор благодарит оргкомитет Турнира Городов и лично Сергея Дориченко за включение в состав варианта осеннего тура 2005 года задачи, возникшей в ходе работы над результатами диссертации.

Наконец, автор глубоко признателен своей семье за постоянную поддержку.

Глава 2

Обзор основных понятий

2.1 Колмогоровская сложность

Понятие колмогоровской (алгоритмической) сложности — один из способов формализации понятия «количество информации». В отличие от энтропии Шеннона, колмогоровская сложность измеряет количество информации не в случайных величинах, а в конечных двоичных словах. В этом разделе мы приведём строгие определения всех понятий и основные факты, которыми будем пользоваться в дальнейшем.

Определение 2.1. Пусть задана некоторая вычислимая функция $V\colon \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$. *Условной колмогоровской сложностью* слова x относительно слова y при способе описания V называется длина кратчайшего слова p, такого что $V(p,y) = x$. Это число будем обозначать $\mathrm{C}_V(x|y)$.

Среди всех способов описания существует асимптотически оптимальный, как показывает следующая теорема:

Теорема 2.2 ([6]). *Существует такой способ описания U, что для любого другого способа описания V существует число c, такое что при всех x и y выполнено неравенство* $\mathrm{C}_U(x|y) < \mathrm{C}_V(x|y) + c$.

Во всех дальнейших определениях и утверждениях предполагается, что некоторый асимптотически оптимальный способ описания зафиксирован. Поэтому мы опускаем соответствующий индекс и обозначаем условную сложность через $\mathrm{C}(x|y)$. Рассматривают также безусловную сложность:

Определение 2.3. *Колмогоровской сложностью* $\mathrm{C}(x)$ слова x называется условная сложность $\mathrm{C}(x \mid \varepsilon)$, где ε — пустое слово.

В литературе встречаются другие варианты определения колмогоровской сложности: префиксная сложность, монотонная сложность, логарифм априорной вероятности и др. Все эти меры совпадают с обычной сложностью с точностью до аддитивного слагаемого порядка $O(\log n)$, где n — длина x. Результаты Мучника и Зиманда, изучаемые в работе, и так формулируются с точностью до такого аддитивного слагаемого, так что переход к другому варианту сложности оставит эти результаты в силе, лишь изменив точное значение констант в $O(\cdot)$-обозначениях.

Рассматривают также колмогоровскую сложность с ограничением на вычислительные ресурсы. Здесь становится важной конкретная модель вычислений. Будем считать, что все вычисления производятся на детерминированной многоленточной машине Тьюринга с выделенными лентами для входов (только для чтения) и выхода (только для записи); время её работы измеряется как число шагов до прихода в завершающее состояние, а память — как число использованных ячеек на рабочих лентах. В следующих определениях под способом описания будем понимать не просто вычислимую функцию, а машину, её вычисляющую.

Определение 2.4. *Условной колмогоровской сложностью* слова x относительно слова y при способе описания V за время t на памяти s называется длина кратчайшего слова p, такого что $V(p, y) = x$, при этом время работы $V(p, y)$ не превосходит t, а использованная память не больше s. Это число обозначается $\mathrm{C}_V^{s,t}(x|y)$.

Теорема об оптимальном способе описания в этом случае принимает такой вид:

Теорема 2.5 ([28]). *Существует такой способ описания U, что для любого другого способа описания V существуют числа c и d, такие что при всех x и y выполнено неравенство $\mathrm{C}_U^{cs,ct\log t}(x|y) < \mathrm{C}_V(x|y) + d$.*

У сложности с ограничением на ресурсы мы также будем опускать индекс, соответствующий способу описания. Безусловная сложность слова x с ограничением на ресурсы вновь определяется как сложность с пустым условием.

Для сложности с ограничением на ресурсы становятся важными и другие аспекты модели вычислений: при использовании вероятностных и/или недетерминированных вычислений необходимые ресурсы могут существенно уменьшиться. Недетерминированные вычисления мы будем понимать

следующим образом: машина получает два параметра (вход y и сертификат q) и возвращает либо специальный символ ошибки $\perp$, либо какое-то слово x. Будем считать, что результат работы машины на входе y равен x, если она возвращает это x для некоторого сертификата q, а для остальных сертификатов возвращает либо то же самое x, либо $\perp$. Время работы машины и использованную память будем считать как максимум этих величин по всем сертификатам.[1] В дальнейшем при рассмотрении недетерминированных машин аргумент q будем опускать. По умолчанию все машины будем считать детерминированными.

Вероятностные вычисления будем понимать следующим образом: помимо стандартных лент, машина имеет отдельную ленту только для чтения, заполненную случайными битами (в бесконечном количестве). По мере надобности машина считывает и использует случайные биты. Время работы на данном входе будем считать как максимальное время по всем случайным битам. Ясно, что общее число прочитанных случайных битов заведомо не превысит времени работы, поэтому результат зависит только от некоторого конечного участка случайной ленты, который мы будем обозначать за r.

Дадим определения трёх сложностей для разных моделей вычислений. Во всех случаях переходы к универсальному способу описания и безусловной сложности проводятся аналогично предыдущему.

Определение 2.6. *Недетерминированной колмогоровской сложностью* слова x относительно слова y при недетерминированном способе описания W за время t на памяти s называется длина кратчайшего слова p, такого что $W(p, y) = x$, при этом время работы $W(p, y)$ не превосходит t, а использованная память не больше s. Это число будем обозначать через $\mathrm{CN}_W^{s,t}(x|y)$.

Определение 2.7. *Вероятностной колмогоровской сложностью* слова x относительно слова y при способе описания V за время t на памяти s называется длина кратчайшего слова p, такого что $\Pr_r[V(p, y, r) = x] > \frac{2}{3}$, при этом время работы $V(p, y, r)$ не превосходит t, а использованная память не больше s при всех r. Это число будем обозначать через $\mathrm{CBP}_V^{s,t}(x|y)$.

[1] Чтобы эти максимумы были конечными, нужно чтобы либо длина сертификата была ограничена некоторой функцией от длины y, либо машина не читала слишком длинные сертификаты полностью. Это стандартное предположение для недетерминированных вычислений.

Определение 2.8. *Сложностью Артура-Мерлина* слова x относительно слова y при недетерминированном способе описания W за время t на памяти s называется длина кратчайшего слова p, такого что $\Pr_r[W(p,y,r)=x] > \frac{2}{3}$, при этом время работы $W(p,y,r)$ не превосходит t, а использованная память не больше s при всех r. Это число будем обозначать через $\mathrm{CAM}_W^{s,t}(x|y)$.

Название последней сложности восходит к Бабаю [11] и объясняется следующей метафорой: вычисления проводит Артур с ограниченными вычислительными ресурсами, но умеющий кидать монетку, при помощи мага Мерлина, дающего подсказку q. Как только Артур бросил монетку столько раз, сколько ему нужно, Мерлин магическим образом узнаёт результаты бросков и может выбрать q. Это соответствует записи $W(p,y,r)$: недетерминированный способ описания получает r в качестве аргумента, поэтому сертификат q может зависеть от r. Нужно, чтобы условие $W(p,y,r)=x$ было выполнено с вероятностью хотя бы $\frac{2}{3}$, т.е. не выполнено с вероятностью не больше $\frac{1}{3}$. Это значит, что Мерлин может обмануть Артура (т.е. может каким-то сообщением заставить его породить слово, отличное от x) или не может убедить Артура (т.е. никаким сообщением не может заставить породить слово x) с суммарной вероятностью не больше $\frac{1}{3}$.

Для сложности с ограничением на ресурсы появляется неэквивалентное (для детерминированных машин) исходному понятие сложности различения. Вместо способов описания мы будем говорить о разрешающих алгоритмах, т.е. машинах, возвращающих 0 или 1.

Определение 2.9 ([43]). *Условной сложностью различения* $\mathrm{CD}_V^{s,t}(x|y)$ называется длина кратчайшего слова p, такого что $V(p,x,y)=1$ и $V(p,z,y)=0$ при $z \neq x$, при этом при всех z программа $V(p,z,y)$ работает время t и использует память s.

Аналогично предыдущему определяется безусловная сложность различения. Также верна теорема об оптимальном разрешающем алгоритме, аналогичная теореме 2.5, поэтому в дальнейшем мы будем опускать индекс, соответствующий разрешающему алгоритму. Можно дать определения сложностей различения для других вычислительных моделей, но в нашей работе они не понадобятся (а для недетерминированных вычислений они будут эквивалентны обычной сложности).

Ясно, что сложность различения не больше сложности порождения: если машина может породить нужное слово, то она может и проверить, оно ли поступило на вход.

Нам также потребуется сложность различения с оракулом:

Определение 2.10. *Условной сложностью различения* $\mathrm{CD}_V^{s,t,A}(x|y)$ с оракулом A называется длина кратчайшего слова p, такого что $V^A(p,x,y)=1$ и $V^A(p,z,y)=0$ при $z\neq x$. Здесь программа V может делать запросы к оракулу A, т.е. за один шаг выяснять про любое слово, лежит ли оно в множестве A, при этом V работает не дольше t шагов и занимает память не больше s.

Имеет место следующая верхняя оценка из [13] (похожая лемма встречалась в работе Сипсера [43]) на CD^A:

Теорема 2.11 ([13])**.** *Пусть $A\subset\{0,1\}^*$. Введём обозначение $A^{=n}=A\cap\{0,1\}^n$. Тогда для некоторых полиномов $t(n)$ и $s(n)$ при всех $x\in A^{=n}$ выполнено $\mathrm{CD}^{s(n),t(n),A^{=n}}(x)\leqslant 2\log|A^{=n}|+O(\log n)$. Более того, соответствующая программа обращается к оракулу только один раз и задаёт вопрос относительно своего входа.*

Нам понадобится следующее следствие:

Следствие 2.12. *Для некоторых полиномов $t(n)$ и $s(n)$ при всех $A\subset\{0,1\}^*$ и $x\in A^{=n}$ существует программа длины не более $2\log|A^{=n}|+O(\log n)$, работающая не более $t(n)$ шагов, использующая память не больше $s(n)$, принимающая x и отвергающая все остальные элементы $A^{=n}$. При этом программа может как принимать, так и отвергать слова, не лежащие в $A^{=n}$.*

Доказательство. Внесём в программу, существующую по теореме 2.11, следующее изменение: вместо запроса к оракулу, принадлежит ли её вход к A, она будет по умолчанию считать, что её вход лежит в A. Тогда оракул более не понадобится, при этом на словах из $A^{=n}$ результат будет таким же, а это и требуется. □

2.2 Экстракторы

Многие вычислительные задачи решаются вероятностными алгоритмами более эффективно, чем детерминированными. Однако, вероятност-

ные алгоритмы требуют источника случайных битов, который может отсутствовать, быть дорогим или несовершенным. Чтобы уменьшить потребность в случайных битах, ищут способы дерандомизировать алгоритм [10; 25; 26; 31]. Одна из идей дерандомизации [46] состоит в том, чтобы взять «не очень равномерно» распределённые случайные биты и преобразовать их в «почти равномерные». Такие преобразования называют экстракторами. Исходная идея может уточняться различными способами (наиболее полные обзоры приведены в [40; 41; 50]), мы сформулируем два наиболее часто встречающихся и используемых варианта. Начнём с формальных определений степени равномерности распределения. Мы будем использовать различные формализации для исходного и преобразованного распределений.

Определение 2.13. *Мин-энтропией* случайной величины ξ, принимающей значения на множестве X, называется максимальное число k, такое что для любого $x \in X$ вероятность того, что $\xi = x$, не превышает $\frac{1}{2^k}$. Обозначение: $H_\infty(\xi)$.

Иными словами, $H_\infty(\xi) = -\log\bigl(\max_{x\in X}\{\Pr[\xi = x]\}\bigr)$.

Определение 2.14. *Статистическим расстоянием* между случайными величинами ξ и η, принимающими значения на множестве X называется величина $\max_{S\subset X}\bigl|\Pr[\xi \in S] - \Pr[\eta \in S]\bigr|$. Эта же величина равна $\frac{1}{2}\sum_{x\in X}\bigl|\Pr[\xi = x] - \Pr[\eta = x]\bigr|$. Обозначение: $\mathrm{dist}(\xi, \eta)$.

Нетрудно заметить, что любое детерминированное преобразование случайной величины не увеличивает её мин-энтропию и не уменьшает расстояние до равномерного распределения. Поэтому для получения распределения, близкого к равномерному, нужно хотя бы два независимых источника случайности. Мерой качества источника случайности (т.е. случайной величины) будем считать её мин-энтропию. Напомним, что через U_l мы обозначаем случайную величину, распределённую равномерно на $\{0,1\}^l$.

Определение 2.15 ([35]). Назовём (k, ε)-экстрактором с одним источником функцию $\mathrm{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, такую что для любой случайной величины ξ на $\{0,1\}^n$ с мин-энтропией не меньше k индуцированная величина $\mathrm{Ext}(\xi, U_d)$ расположена на расстоянии не больше ε от U_m. (Здесь предполагается, что распределение U_d, U_m и ξ независимы).

Разумеется, формально у этого экстрактора два источника случайности. Но второй из них фиксирован, что позволяет рассматривать такой экстрактор как случайное преобразование одной случайной величины. В англоязычной литературе такой объект называют seeded extractor («экстрактор со случайным зерном»). Если исходных источников случайности хотя бы два, то необходимость в случайном зерне отпадает.

Определение 2.16. Назовём (k, ε)-экстрактором с двумя источниками функцию $\mathrm{Ext}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, такую что для любых независимых случайных величин ξ_1 и ξ_2, таких что $H_\infty(\xi_1) \geqslant k$ и $H_\infty(\xi_2) \geqslant k$, выполнено $\mathrm{dist}(\mathrm{Ext}(\xi_1, \xi_2), U_m) < \varepsilon$.

Можно рассматривать экстракторы и с бóльшим числом источников, но нам они не понадобятся.

Функцию двух аргументов, в частности экстрактор с одним источником, можно эквивалентно рассматривать как двудольный граф с кратными рёбрами и одинаковыми степенями всех вершин левой доли. В таком случае первый аргумент задаёт вершину левой доли, второй аргумент — номер исходящего ребра, а значение — вершину правой доли. Для двудольных графов определяющее свойство экстрактора можно переформулировать следующим образом.

Определение 2.17. Двудольный граф с кратными рёбрами (L, R, E), где L — левая доля, состоящая из N элементов, R — правая доля, состоящая из M элементов, а E — множество рёбер, т.е. мультимножество пар из $L \times R$, в котором у каждой вершины левой доли степень равна D, называется (K, ε)-экстрактором, если для любого множества $S \subset L$, такого что $|S| \geqslant K$, и любого множества $Y \subset R$ выполнено:

$$\left| \frac{|Y|}{|R|} - \frac{|E(S,Y)|}{|E(S,R)|} \right| < \varepsilon, \tag{2.1}$$

где через $E(P, Q)$ обозначено множество $E \cap (P \times Q)$, т.е. множество рёбер, ведущих из вершин множества P в вершины множества Q. Иными словами, доля выходящих из S рёбер, идущих в множество Y, отличается от доли вершин, лежащих в множестве Y, не больше, чем на ε.

Определения графа-экстрактора и функции-экстрактора (с одним источником) эквивалентны в следующем смысле:

Утверждение 2.18 (Неявно в [16]). *Если функция* Ext *является* (k,ε)*-экстрактором, то соответствующий ей граф является* $(2^k,\varepsilon)$*-экстрактором. Если* $N = 2^n$, $M = 2^m$ *и* $D = 2^d$, *где* n, m, d *суть целые числа, а граф является* (K,ε)*-экстрактором, то соответствующая функция является* $(\log K,\varepsilon)$*-экстрактором.*

В дальнейшем будем называть двудольным графом типа (n, m, d) граф с кратными рёбрами, имеющий $N = 2^n$ врешин в левой доле, $M = 2^m$ вершин в правой доле и степень каждой вершины слева $D = 2^d$.

Сформулированное утверждение следует из леммы 5 в [16], но для полноты изложения мы приведём его прямое доказательство, которое автор узнал от Максима Бабенко. Также стоит отметить, что на похожей идее основана 6-ая задача осеннего тура двадцать седьмого Турнира Городов за 8–9 классы [7].[2]

Доказательство. Свойство графа-экстрактора следует из свойства функции-экстрактора для распределения, равномерного на множестве S. Поэтому в одну сторону утверждение очевидно. В другую сторону нужно доказать, что если свойство функции-экстрактора выполнено для всех равномерных распределений на множествах размера 2^k, то оно выполнено для всех распределений с мин-энтропией не меньше k. Для этого докажем, что любое такое распределение можно представить в виде выпуклой комбинации равномерных распределений на множествах размера $K = 2^k$. Более того, в этой комбинации будет не больше N ненулевых коэффициентов.

Переформулируем последнее утверждение следующим образом: пусть дано N неотрицательных чисел, каждое из которых не превышает $1/K$ от общей суммы. Назовём «операцией» одновременное уменьшение K из этих чисел на одну и ту же величину («величину операции»), при котором все числа остаются неотрицательными и никакое из них по-прежнему не превышает $1/K$ от общей суммы. Тогда не более чем за N операций можно обратить все числа в 0.

Докажем сформулированное утверждение методом математической индукции. База ($N = K = 1$) очевидна. В процессе перехода будем уменьшать на единицу либо и N, и K, либо только N. Выберем произвольные

[2]Также автору известно остроумное доказательство, использующее выпуклый анализ, описанное Ильёй Ирхиным [4] по мотивам лекций Дмитрия Ицыксона [5].

K ненулевых чисел из набора. Уменьшим их на максимально возможную величину, при которой не нарушаются условия. Возможны два случая. Во-первых, одно из выбранных чисел может обратиться в нуль, тогда можно его исключить и применить предположение индукции для $N-1$ и K. Действительно, каждое из чисел по-прежнему не больше $1/K$ от общей суммы, и для оставшихся чисел достаточно $N-1$ операции. Во-вторых, одно из невыбранных чисел (обозначим его за x) может стать равным $1/K$ от общей суммы. (Ясно, что ни с каким из выбранных чисел такого случиться не может, поскольку они уменьшаются, а невыбранные — нет). В таком случае сумма всех чисел, кроме x, равна $(K-1)/K$ общей суммы, а каждое из чисел не превышает $\frac{1}{K}/\frac{K-1}{K} = 1/(K-1)$ суммы всех чисел, кроме x. Таким образом, к набору чисел без x можно применить предположение индукции для $N-1$ и $K-1$. К каждому набору чисел, к которому применяется операция из предположения индукции, мы добавим число x, сделав таким образом из $K-1$ числа K чисел. Сумма величин всех этих операций равна $1/(K-1)$ от суммы всех чисел, кроме x, т.е. $1/(K-1)\cdot(K-1)/K = 1/K$ от суммы всех чисел, т.е. равна x. Таким образом, при добавлении x ко всем наборам после проведения всех операций все числа обнулятся, что и требовалось. □

Вероятностным методом [1] можно доказать существование экстракторов с очень хорошими параметрами. А именно, верна следующая теорема.

Теорема 2.19 ([36]). *При всех n, k и ε существуют экстракторы для $d = \log(n-k) + 2\log(1/\varepsilon) + O(1)$ и $m = k + d - \log(1/\varepsilon) - O(1)$.*

Однако для приложений желательно, чтобы экстракторы задавались явной конструкцией. Формально под явными экстракторами понимаются вычисляемые за полиномиальное время:

Определение 2.20. Пусть $m(n)$, $d(n)$, $k(n)$ и $\varepsilon(n)$ суть вычислимые за полиномиальное время функции натурального аргумента. (Значения первых трёх — также натуральные числа, последней — рациональные числа на $(0,1)$). Тогда семейство функций $\mathrm{Ext}_n\colon \{0,1\}^n \times \{0,1\}^{d(n)} \to \{0,1\}^{m(n)}$ называется семейством явных (k,ε)-экстракторов (или просто явным (k,ε)-экстрактором), если Ext_n вычисляется за полиномиальное от n время и при всех n является $(k(n),\varepsilon(n))$-экстрактором.

За последние 20 лет построено множество различных явных экстракторов, но ни одна из конструкций [17; 21; 29; 33; 37; 38; 48; 49] не даёт оптимальных параметров. Нам потребуются экстракторы, существование которых утверждает следующая теорема:

Теорема 2.21 ([37]). *Для всех n, k, m и $\varepsilon > 0$, таких что $m \leqslant k \leqslant n$, существуют явные (k, ε)-экстракторы для $d = O\left(\frac{\log^2(n/\varepsilon)}{\log(k/m)}\right)$ и для $d = O(\log^2(n/\varepsilon) \cdot \log(1/\gamma))$, где $\gamma = \frac{k-m+1}{m-1} < \frac{1}{2}$.*

Мы воспользуемся этой теоремой для таких параметров, из которых выводится

Следствие 2.22. *Для всех n, $k \leqslant n$ и $\varepsilon > 1/\operatorname{poly}(n)$ существуют явные (k, ε)-экстракторы для $m = k$ и $d = O(\log^3 n)$.*

Доказательство. Воспользуемся теоремой 2.21 для $m = k$ и $\varepsilon = 1/\operatorname{poly}(n)$. Тогда $\gamma = \frac{1}{m-1} = \frac{1}{k-1}$ и $d = O(\log^2(n/\varepsilon) \cdot \log(1/\gamma)) = O(\log^2(\operatorname{poly}(n)) \cdot \log(k-1)) = O(\log^3 n)$. □

Экстракторы с двумя источниками, как и экстракторы с одним источником, удобно рассматривать не как функции на словах, а как комбинаторные объекты, а именно раскраски квадратных таблиц:

Определение 2.23. Раскраску таблицы $A \times A$ в множество цветов C, т.е. функцию $\operatorname{Col}\colon A \times A \to C$, назовём (K, ε)-экстракторной, если для любых множеств $S \subset A$ и $T \subset A$ размера не меньше K и любого множества $Y \subset C$ выполнено

$$\left| \frac{|Y|}{|C|} - \frac{|\operatorname{Col}^{-1}(Y) \cap (S \times T)|}{|S| \cdot |T|} \right| < \varepsilon, \tag{2.2}$$

иными словами, доля клеток прямоугольника $S \times T$, раскрашенных в цвета из Y, отличается от доли цветов из Y среди всех цветов не больше, чем на ε.

Соответствие между функциями и раскрасками таблиц строится естественным образом, и можно доказать по аналогии с утверждением 2.17 теорему об эквивалентности двух определений.

Также рассматриваются следующие усиления понятий экстрактора:

Определение 2.24. Функция $\operatorname{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ называется (k, ε)-экстрактором с одним источником в сильном смысле, если для любой

случайной величины ξ на $\{0,1\}^n$ с минимальной энтропией не меньше k и случайной величины η, распределённой равномерно на $\{0,1\}^d$ независимо от ξ, индуцированная величина $(\mathrm{Ext}(\xi,\eta),\eta)$ расположена на расстоянии не больше ε от U_{m+d}.

Интуитивный смысл усиления таков: экстрактор извлекает случайность именно из слабого источника, а не из предоставленных ему истинно случайных битов.

Определение 2.25. Функция $\mathrm{Ext}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ называется (k,ε)-экстрактором с двумя источниками в сильном смысле, если для любых независимых случайных величин ξ_1 и ξ_2, таких что $H_\infty(\xi_1) \geqslant k$ и $H_\infty(\xi_2) \geqslant k$, выполнено $\mathrm{dist}((\xi_1,\mathrm{Ext}(\xi_1,\xi_2)),(\xi_1,U_m)) < \varepsilon$ и $\mathrm{dist}((\mathrm{Ext}(\xi_1,\xi_2),\xi_2),(U_m,\xi_2)) < \varepsilon$, где распределение U_m не зависит от ξ_1 и ξ_2.

Здесь интуитивный смысл состоит в том, что экстрактор «производит новую случайность», не зависящую ни от одного из начальных источников.

2.3 Колмогоровские экстракторы

Колмогоровские экстракторы — иной способ формализации процедуры, «извлекающей случайность» из неравномерного источника случайности. Под объектами, «содержащими случайность», понимаются теперь не распределения, а конечные слова, а под мерой случайности — колмогоровская сложность: чем ближе сложность слова к его длине, тем более слово случайно. Как и для обычных экстракторов, одного источника недостаточно для «извлечения случайности»: любая (всюду определённая) вычислимая процедура не уменьшает дефект случайности (т.е. разность между длиной и сложностью) для какого-то входа. Поэтому источников случайности должно быть хотя бы два. С другой стороны, требование независимости можно ослабить и требовать лишь ограниченной зависимости.

Определение 2.26. *Зависимостью* между строками x и y называется величина $\mathrm{dep}(x,y) = \mathrm{C}(x) + \mathrm{C}(y) - \mathrm{C}(x,y)$.

Некоторые авторы используют другие определения зависимости, отличающиеся от исходного на слагаемое порядка $O(\log n)$, например:

Утверждение 2.27 (Теорема Колмогорова–Левина, см. [2], теорема 21)**.** $\mathrm{dep}(x,y) = \mathrm{C}(x) - \mathrm{C}(x|y) + O(\log n) = \mathrm{C}(y) - \mathrm{C}(y|x) + O(\log n)$.

Также зависимость часто называют взаимной информацией и обозначают $I(x:y)$.

Теперь определим понятие колмогоровского экстрактора. Оно будет аналогом определения экстрактора с двумя источниками.

Определение 2.28. Функция $\mathrm{KExt}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ называется (k,α,d)-колмогоровским экстрактором, если для всех x и y, таких что $\mathrm{C}(x) > k$, $\mathrm{C}(y) > k$ и $\mathrm{dep}(x,y) < \alpha$, выполнено $\mathrm{C}(\mathrm{KExt}(x,y)) > m - d$.

Доказано, что свойства экстрактора с двумя источниками и колмогоровского экстрактора очень близки. Более конкретно, можно дать определение «приближённого экстрактора» (almost extractor) в терминах вероятностных распределений и мин-энтропии, которое будет эквивалентно определению колмогоровского экстрактора с точностью до небольшого ухудшения параметров при переходе в каждую сторону. За подробностями отсылаем читателя к оригинальным работам [18; 22], а также обзору [56].

Определение колмогоровского экстратора также можно усилить:

Определение 2.29. Функция $\mathrm{KExt}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ называется (k,α,d)-колмогоровским экстрактором в сильном смысле, если для всех x и y, таких что $\mathrm{C}(x) > k$, $\mathrm{C}(y) > k$ и $\mathrm{dep}(x,y) < \alpha$, выполнено $\mathrm{C}(\mathrm{KExt}(x,y)|x) > m - d$ и $\mathrm{C}(\mathrm{KExt}(x,y)|y) > m - d$.

Построение колмогоровских экстракторов, в том числе в сильном смысле, описано в работах Мариуса Зиманда [53–55]. Конструкции Зиманда основаны на параллелизме между сложностными и комбинаторными свойствами функций. Приведём формулировки теорем, полученных Зимандом:

Теорема 2.30 ([54], теорема 3.1, и [55], теорема 4)**.** *Пусть $k(n)$ и $\alpha(n)$ суть вычислимые функции, такие что $1 < k(n) < n$ и $1 < \alpha(n) < k(n) - O(\log n)$. Тогда существуют $(k(n), \alpha(n), \alpha(n) + O(\log n))$-колмогоровский экстрактор для $m = 2k(n) - O(\log n)$ и $(k(n), \alpha(n), \alpha(n) + O(\log n))$-колмогоровский экстрактор в сильном смысле для $m = k(n) - O(\log n)$.*

В главе 6 мы обобщим понятие колмогоровского экстрактора на случай сложности с ограничением на память и переложим доказательство Зиманда на этот случай. Там же мы опишем основной комбинаторный инстру-

мент, введённый Зимандом и использованный им для доказательства — пёстрые и равномерно пёстрые таблицы.

2.4 Схемы из функциональных элементов

Схемы из функциональных элементов давно изучаются в теории сложности вычислений. Разные авторы используют слегка отличные определения, поэтому мы уточним определение, а также приведём формулировки теорем, которые нам потребуются.

Схемой из функциональных элементов будем называть ориентированный граф без циклов, каждая вершина которого помечена одной из меток i, o, $\neg$, $\wedge$, $\vee$. Накладываются следующие условия: вершины с метками i не имеют входящих рёбер, вершины с метками o или $\neg$ имеют ровно одно входящее ребро, вершины с метками $\wedge$ или $\vee$ имеют хотя бы одно входящее ребро, вершины с метками o не имеют исходящих рёбер, а вершины с метками $\neg$, $\wedge$, $\vee$ имеют хотя бы одно исходящее ребро. Отсутствие циклов позволяет корректно определить *высоту* каждой вершины как максимальную длину пути, идущего из одной из вершин с меткой i в данную. Если вершинам с метками i присвоить булевы значения, то значения в любой другой вершине можно посчитать, применив функцию, соответствующую её метке, к значениям в вершинах, из которых в неё идёт ребро. (Такие вершины будем называть прообразами данной). При этом метке o соответствует тождественная функция, т.е. в неё переносится значение единственного прообраза, а конъюнкции и дизъюнкции применяются ко всем прообразам. Таким образом, схема с n вершинами типа i и k вершинами типа o вычисляет некоторую функцию $f\colon \{0,1\}^n \to \{0,1\}^k$. *Размером* схемы будем называть количество вершин в ней, а *глубиной* — максимальную высоту вершины.

Одной из ключевых тем теории сложности вычислений является получение верхних и нижних оценок на размер или глубину схем, которые вычисляют ту или иную конкретную функцию. В частности, была доказана теорема о невозможности вычисления схемами полиномиального размера и константной глубины функции большинства и других пороговых функций.

Теорема 2.31 ([19]). *Для любого многочлена $p(\cdot)$ и любой константы d существует n, для которого никакая схема размера $p(n)$ и глубины d не вычисляет верно функцию* $\mathrm{maj}\colon \{0,1\}^n \to \{0,1\}$, *значение которой*

совпадает с большинством аргументов[3]*. То же утверждение верно при любом (фиксированном)* $\alpha \in (0,1)$ *для функции* $\mathrm{thr}_\alpha \colon \{0,1\}^n \to \{0,1\}$*, равной 1, если вес её входной строки (т.е. доля единиц среди аргументов) больше* α*.*

Вместе с тем, существуют схемы константной глубины для приближённого вычисления пороговых функций [8; 52]. Мы будем использовать следующую теорему:

Теорема 2.32 ([52])**.** *Для любого целого* $d \geqslant 3$ *и любого* $\varepsilon = \Omega(1/\log^{d-3} n)$ *существует схема размера* $\mathrm{poly}(n)$ *и глубины* d*, возвращающая* 1 *на входах веса* $1/2 + \varepsilon$ *и* 0 *на входах веса* $1/2 - \varepsilon$*.*

Если же порог α не фиксирован, а достаточно быстро стремится к нулю с ростом n, то пороговую функцию можно вычислить точно:

Теорема 2.33 ([8])**.** *Для любых* n *и* i *существует схема глубины* d *и размера* $\mathrm{poly}(n)$*, возвращающая* 1 *на входах длины* n*, содержащих меньше* $\log^i n$ *единиц, и возвращающая 0 на остальных входах. При этом глубина* d *не зависит от* n*, но может зависеть от* i*.*

2.5 Генераторы псевдослучайных чисел

Генераторы псевдослучайных чисел — ещё один объект, формализующий интуитивное понятие «процедура, производящая новую случайность». Исторически эти объекты стали изучаться первыми, но существование многих видов генераторов доказано лишь при условии верности тех или иных теоретико-сложностных предположений, например, о существовании односторонней функции [24]. Нисан и Вигдерсон [34] доказали существование генераторов определённого вида, не опираясь на какие-либо предположения, и мы воспользуемся этой конструкцией сразу в двух главах: 5 и 6.

Определение 2.34. Пусть для каждого n задан класс $\mathcal{D}_n$ функций-отличителей, отображающих $\{0,1\}^n$ в $\{0,1\}$. Семейство функций $G_n \colon \{0,1\}^{k(n)} \to \{0,1\}^n$ называется *генератором псевдослучайных чисел* для класса отличителей $\mathcal{D}_n$, если для любого многочлена $p(\cdot)$ при всех достаточно больших n для всех функций D_n из $\mathcal{D}_n$ выполнено

$$\left|\Pr_{x\in\{0,1\}^{k(n)}}\left[D_n(G_n(x)) = 1\right] - \Pr_{y\in\{0,1\}^n}\left[D_n(y) = 1\right]\right| < \frac{1}{p(n)}.$$

[3]Значение может быть любым при равенстве количеств нулей и единиц среди аргументов.

Как правило, в качестве класса $\mathcal{D}_n$ берут либо полиномиально вычислимые функции, либо функции, вычислимые схемами полиномиального размера. Однако, в таком случае существование генераторов удаётся доказать только с опорой на различные теоретико-сложностные гипотезы. Безусловный результат получается, если рассмотреть класс функций, вычислимых схемами полиномиального размера и константной глубины.

Теорема 2.35 ([32; 34]). *Для любой константы d и любого положительного полинома $q(n)$ существует семейство функций $G_n\colon \{0,1\}^k \to \{0,1\}^n$, где $k = O(\log^{2d+6} n)$, такое что:*

- *Функция G_n вычислима на памяти $\mathrm{poly}(k)$;*[4]
- *Функция G_n является генератором псевдослучайных чисел для класса всех функций, вычислимых схемами размера $q(n)$ и глубины не больше d.*

Заменив переменные, можно получить следующее следствие, используемое в дальнейшем. Чтобы исключить разночтения, мы запишем все упомянутые в нём полиномы явно.

Следствие 2.36. *Для любой константы d и любых положительных полиномов $p(n)$ и $q(n)$ существуют полиномы $r(n)$ и $s(n)$ и семейство функций $G_n\colon \{0,1\}^{r(n)} \to \{0,1\}^{2^{p(n)}}$, такие что:*

- *Функция G_n вычислима на памяти $s(n)$;*
- *Функция G_n является генератором псевдослучайных чисел для класса всех функций, вычислимых схемами размера не больше $2^{q(n)}$ и глубины не больше d.*

Доказательство. Напрямую применить теорему 2.35 не получится, из-за того что при $q(n) > np(n)$ величина $2^{q(n)}$ не будет полиномом от $2^{p(n)}$. Вместо этого мы заметим следующее: во-первых, случай $q(n) < p(n)$ тривиален, поскольку мы учитываем входные элементы в размере схемы, так что никакая схема размера $2^{q(n)}$ не сможет получить на вход слово длины $2^{p(n)}$. Во-вторых, по теореме 2.35 существуют полиномы $r'(n)$ и $s'(n)$ и семейство функций $G'_n\colon \{0,1\}^{r'(n)} \to \{0,1\}^{2^{q(n)}}$, такие что G'_n вычислима на

[4]Напомним, что мы считаем память только на рабочих лентах машины Тьюринга. Иначе говоря, для любого x любой конкретный бит значения $G_n(x)$ можно вычислить на памяти $\mathrm{poly}(k)$, хотя общая длина $G_n(x)$ экспоненциальна от $k = |x|$.

памяти $s'(n)$ и является генератором псевдослучайных чисел для класса всех функций, вычислимых схемами размера не больше $2^{q(n)}$ и глубины не больше d. Действительно, потребуется случайное зерно длины порядка $\big(\log(2^{q(n)})\big)^{2d+6} = q(n)^{2d+6}$, что является полиномом. В частности, функция G'_n будет генератором псевдослучайных чисел для класса всех функций, вычислимых такими схемами и зависящих только от первых $2^{p(n)}$ битов слова (это корректно, поскольку $p(n) \leqslant q(n)$). Таким образом, в качестве G_n можно взять префикс длины $2^{p(n)}$ функции G'_n, а полиномы взять те же: $r(n) = r'(n) = q(n)^{2d+6}$, $s(n) = s'(n)$. □

Отсюда выводится следующий базовый принцип:

Лемма 2.37. *Пусть $\mathcal{C}_n$ есть некоторое множество комбинаторных объектов, закодированных строками длины $2^{\mathrm{poly}(n)}$. Пусть $\mathcal{P}_n$ есть свойство, выполненное хотя бы для (константной) доли $\alpha > 0$ объектов из $\mathcal{C}_n$, где $\alpha > 0$ есть некоторая константа, при этом это свойство можно распознать схемами размера $2^{\mathrm{poly}(n)}$ и константной глубины. Тогда при всех достаточно больших n свойство $\mathcal{P}_n$ выполнено для доли не меньше $\alpha/2$ значений G_n, где G_n — функция из предыдущего следствия.*

Доказательство. Согласно следствию 2.36, функция G_n «обманывает» все схемы размера $2^{\mathrm{poly}(n)}$ и константной глубины, в частности схемы, распознающие $\mathcal{P}_n$. Раз доля объектов, удовлетворяющих свойству $\mathcal{P}_n$, среди всех объектов не меньше α, то при достаточно больших n доля объектов, удовлетворяющих свойству $\mathcal{P}_n$, среди значений G_n достаточно близка к α, хотя бы больше $\alpha/2$. □

2.6 Вычислительная XOR-Лемма

В литературе встречается несколько утверждений, которые получили название «XOR-лемма». Нам потребуется так называемая вычислительная XOR-Лемма, доказанная Вазирани и Вазирани в работе [51]. Эта лемма связывает вероятность предсказания псевдослучайной величины и успешность отличителя этой псевдослучайной величины от истинно случайной. Мы сформулируем эту лемму в усечённой форме, полную формулировку и обзор остальных XOR-лемм можно найти в обзоре [20].

Теорема 2.38 (Вычислительная XOR-лемма [51], частный случай)**.** *Пусть z есть случайная величина, равномерно распределённая на $\{0,1\}^m$, а σ есть случайный бит (т.е. случайная величина, равномерно распределённая на $\{0,1\}$). Пусть $f\colon \{0,1\}^m \to \{0,1\}^k$ и $g\colon \{0,1\}^m \to \{0,1\}$ суть полиномиально вычислимые функции. Пусть $T\colon \{0,1\}^k \times \{0,1\} \to \{0,1\}$ — некоторый вероятностный полиномиальный алгоритм, причём*

$$\Pr_z\left[T(f(z),g(z))=1\right] - \Pr_{z,\sigma}\left[T(f(z),\sigma)=1\right] > \varepsilon. \tag{2.3}$$

Пусть $G_\sigma\colon \{0,1\}^k \to \{0,1\}$ — алгоритм, определённый равенством $G_\sigma(x) = T(x,\sigma)\oplus\sigma\oplus 1$. (Где $\oplus$ обозначает сложение по модулю 2). Тогда

$$\Pr_{z,\sigma}\left[G_\sigma(f(z)) = g(z)\right] > \frac{1}{2} + \varepsilon. \tag{2.4}$$

2.7 Отдельные используемые неравенства

В этом разделе мы приведём список неравенств, которые будут использоваться в выкладках.

Вначале докажем несложную оценку для числа сочетаний. Общеизвестно, что $C_n^k = \frac{n(n-1)\ldots(n-k+1)}{k!}$. Оценив каждый множитель в числителе числом n, получим, что $C_n^k \leqslant \frac{n^k}{k!}$. В силу известной формулы Стирлинга выполнено $k! > \left(\frac{k}{e}\right)^k$, что при подстановке даёт оценку

$$C_n^k < \left(\frac{ne}{k}\right)^k. \tag{2.5}$$

Далее, сформулируем несколько вариантов неравенства Чернова, оценивающего сверху вероятность больших уклонений. Первый вариант обычно называют неравенством Хёфдинга.

Теорема 2.39 ([23])**.** *Пусть $\xi_1,\ldots,\xi_n$ суть независимые одинаково распределённые случайные величины, принимающие значения из $[0,1]$ и имеющие математическое ожидание p. Тогда для любого $\varepsilon > 0$ вероятность события $\sum \xi_i > (p+\varepsilon)n$ не превышает $e^{-2\varepsilon^2 n}$, и вероятность события $\sum \xi_i < (p-\varepsilon)n$ также не превышает $e^{-2\varepsilon^2 n}$.*

Далее, нам потребуется более общая форма, которую обычно называют неравенством Чернова–Хёфдинга:

Теорема 2.40 ([23]). *Пусть $\xi_1, \ldots, \xi_n$ суть независимые одинаково распределённые бернуллиевские случайные величины, имеющие математическое ожидание p. Тогда для любого $\varepsilon > 0$ вероятность события $\sum \xi_i > (p+\varepsilon)n$ не превышает $e^{-D(p+\varepsilon\|p)n}$, где $D(x\|y)$ — расстояние Кульбака–Лейблера между бернуллиевскими случайными величинами, принимающими значение 1 с вероятностями x и y соответственно, т.е.*

$$D(x\|y) = x\ln\frac{x}{y} + (1-x)\ln\frac{1-x}{1-y}. \tag{2.6}$$

Мы будем применять это неравенство для специального случая, когда $\varepsilon = p$. В этом случае верно следующее

Следствие 2.41. *Пусть $\xi_1, \ldots, \xi_n$ суть независимые одинаково распределённые бернуллиевские случайные величины с вероятностью выпадения единицы $p < \frac{1}{2}$. Тогда вероятность события $\sum \xi_i > 2pn$ не превышает $e^{-\frac{1}{3}pn}$.*

Заметим, что применение теоремы 2.39 при малых p дало бы худшую оценку e^{-2p^2n}.

Доказательство. Подставим в формулу (2.6) $2p$ вместо x и p вместо y. Получим величину

$$2p\ln 2 + (1-2p)(\ln(1-2p) - \ln(1-p)). \tag{2.7}$$

Воспользуемся неравенством

$$\ln(1-p) < -p, \tag{2.8}$$

а также формулой Тейлора для функции $(1-2p)\ln(1-2p)$ в форме Лагранжа: $(1-2p)\ln(1-2p) = -2p + 2p^2 + \frac{4p^3}{3(1-\theta)^2}$, откуда

$$(1-2p)\ln(1-2p) > -2p + 2p^2. \tag{2.9}$$

Использовав неравенства (2.8) и (2.9), оценим величину (2.7) величиной $2p\ln 2 + p(1-2p) - 2p + 2p^2 = (2\ln 2 - 1)p > \frac{1}{3}p$. Подставив эту оценку величины $D(2p\|p)$ в утверждение теоремы 2.40, получаем утверждение следствия. Таким образом, следствие доказано. □

Глава 3

Применение экстракторов для доказательства теоремы Мучника об условной сложности

Теорема Ан. А. Мучника ([30]) — одно из главных достижений теории колмогоровской сложности начала XXI века. Неформально она гласит, что для любых слов a и b существует программа p, печатающая a на входе b, одновременно имеющая близкую к минимально возможной длину и простая относительно a. Этот результат очень сильный, поскольку алгоритм, порождающий p из a, практически ничего не знает про b, но при этом закладывает в программу p всю информацию об a, отсутствующую в b. Можно представить себе такую коммуникационную ситуацию: имеются два агента: Алиса, знающая a, и Боб, знающий b. Алиса должна передать Бобу информацию об a, используя канал ограниченной пропускной способности (см. рис. 3.1). Очевидно, что это невозможно сделать алгоритмически, если пропускная способность меньше условной сложности a относительно b. Теорема гласит, что эта оценка точна, если разрешить Алисе и Бобу использовать небольшое число битов «совета», т.е. специально подобранной (и зависящей от a и b) информации, поступающей извне (см. рис. 3.2).

Теорема Мучника устанавливает параллели между различными областями теоретической информатики. Так, утверждение теоремы аналогично теореме Вольфа–Слепяна ([44]) в шенноновской теории информации. Теорема Вольфа–Слепяна формализует ту же самую коммуникационную ситуацию в случае, когда количество информации меряется как шенноновская энтропия, и получает аналогичное необходимое и достаточное условие. Кроме того, различные доказательства теоремы Мучника позволяют связать теоретико-сложностное утверждение с различными комбинаторными

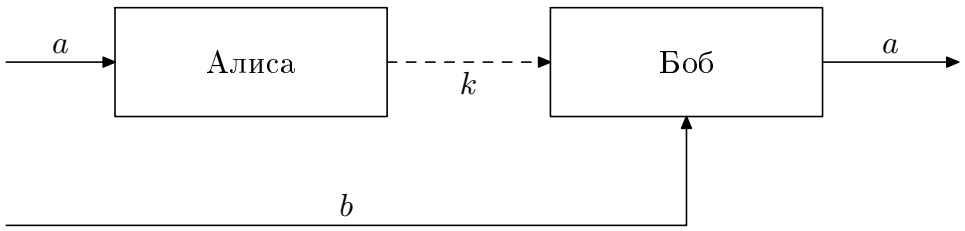


Рис. 3.1: Схема коммуникации: Алиса получает слово a и должна переслать сообщение длиной не больше k битов, так чтобы Боб, получивший слово b, мог восстановить a.

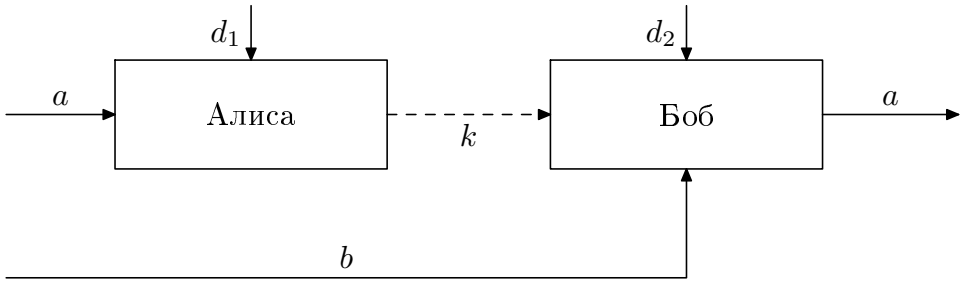


Рис. 3.2: Схема коммуникации с подсказками: Алиса и Боб могут получать короткие подсказки d_1 и d_2, зависящие и от a, и от b.

утверждениями.

В разделе 3.1 мы приведём строгую формулировку теоремы и опишем общую идею доказательства, в том числе сошлёмся на комбинаторные утверждения, связанные со сложностными. В разделе 3.2 мы подробно опишем новую технику доказательства, использующий экстракторы. В разделе 3.3 мы сформулируем и докажем новым способом обобщение теоремы Мучника на случай нескольких условий, при этом в доказательстве будет использоваться понятие префиксного экстрактора, обобщающее понятие обычного.

3.1 Формулировка теоремы и идея доказательства

Мы будем доказывать теорему в следующей формулировке:

Теорема 3.1. *Пусть даны слова из нулей и единиц a и b и числа n и k, такие что $\mathrm{C}(a) < n$ и $\mathrm{C}(a|b) < k$. Тогда существует такое слово p, что:*

- $\mathrm{C}(a|p,b) \leqslant O(\log n)$;
- $\mathrm{C}(p) \leqslant k + O(\log n)$;
- $\mathrm{C}(p|a) \leqslant O(\log n)$,

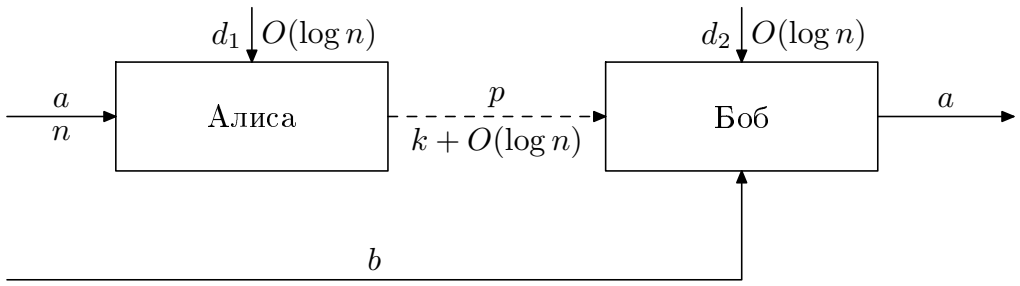


Рис. 3.3: Иллюстрация к теореме Мучника: если Алиса и Боб могут получать подсказки логарифмической длины, зависящие и от a, и от b, то передача информации возможна. Сверху/слева от стрелок подписаны названия слов, а снизу/справа — их длины.

где константы в $O(\cdot)$-обозначениях не зависят от a, b, n, k, но могут зависеть от выбранного способа описания в определении C.

В терминах коммуникации между Алисой и Бобом теорема Мучника утверждает, что коммуникация возможна, если ширина канала составляет $k + O(\log n)$ битов, а Алиса и Боб получают подсказки логарифмической длины (см. рис. 3.3).

Сразу можно сделать несколько несложных замечаний по формулировке:

1. Во втором неравенстве можно заменить сложность программы $\mathrm{C}(p)$ на её длину $|p|$, тем самым усилив формулировку. Действительно, вместо само́й программы можно использовать её кратчайшее описание: первое неравенство останется в силе, поскольку программу можно получить по её кратчайшему описанию, а третье — поскольку кратчайшее описание можно получить, зная программу и её сложность. Если есть несколько кратчайших описаний, то нужно взять лексикографически первое. Двоичная запись сложности имеет длину не больше $\log n$, которая прибавится к правой части и лишь увеличит константу в $O(\cdot)$-обозначении.

2. Можно считать, что $k = \mathrm{C}(a|b) + 1$: это минимально возможное значение k, большее $\mathrm{C}(a|b)$. Если теорема верна для такого k, то для всех бо́льших она верна тем более. Поэтому можно заменить во втором неравенстве $k + O(\log n)$ на $\mathrm{C}(a|b) + O(\log n)$. Таким образом, мы получим неравенство $|p| \leqslant \mathrm{C}(a|b) + O(\log n)$. Аналогично можно положить $n = \mathrm{C}(a) + 1$.

3. Можно отбросить последние $O(\log n)$ битов программы p и дописать их к $O(\log n)$ битам, задающим a при известных p и b. Таким образом, первое неравенство останется в силе. Третье неравенство при этом тоже сохранится. В итоге мы получим следующую эквивалентную формулировку: *для любых двоичных слов a и b существует строка p длины не больше* $\mathrm{C}(a|b)$, *такая что* $\mathrm{C}(a|p,b) \leqslant O(\log \mathrm{C}(a))$ *и* $\mathrm{C}(p|a) \leqslant O(\log \mathrm{C}(a))$.

4. Можно, наоборот, добиться выполнения условия $\mathrm{C}(a|p,b) \leqslant O(1)$. Для этого нужно включить $O(\log n)$ битов, описывающих a при известных p и b в исходной формулировке, в состав программы p и в состав описания p при известном a. При этом, разумеется, длину p уже нельзя будет уменьшить до k, а константа в оценке $\mathrm{C}(p|a)$ возрастёт.

5. По причинам, указанным в первом замечании, достаточно доказать теорему для случая, когда не только сложность, но и длина a меньше n: замена слова a на его кратчайшее описание изменяет все сложности, содержащие a, не более, чем на $O(\log n)$.

Общий метод доказательства естественным образом вытекает из формулировки. Неравенство $\mathrm{C}(p|a) \leqslant O(\log n)$ означает, что слову a каким-то вычислимым образом сопоставлены $2^{O(\log n)} = \mathrm{poly}(n)$ слов (будем называть из образами a), среди которых нужно выбрать p. Неравенство $\mathrm{C}(p) \leqslant k + O(\log n)$ означает, что каждое из этих слов имеет сложность (или длину) не больше $k + O(\log n)$. Задача состоит в построении такой конструкции, чтобы хотя бы для одного из слов p было выполнено неравенство $\mathrm{C}(a|p,b) \leqslant O(\log n)$. Естественный способ этого добиться таков: заметим, что при известном b можно перечислять элементы множества $S_b = \{x \mid \mathrm{C}(x|b) < k\}$. В этом множестве заведомо лежит a. Кроме того, можно перечислять элементы S_b, которым сопоставлено слово p (т.е. прообразы p): слово a также лежит среди них. Если таких элементов полиномиальное количество, то слово a может быть задано своим номером среди них. Таким образом, требуется, чтобы хотя бы у одного образа a было полиномиальное число прообразов внутри S_b.

Мы получили, что для доказательства теоремы Мучника достаточно доказать существование функции $f\colon \{0,1\}^{<n} \times \{0,1\}^d \to \{0,1\}^k$, где $d = O(\log n)$, со следующим свойством: для всех двоичных слов a и b, таких

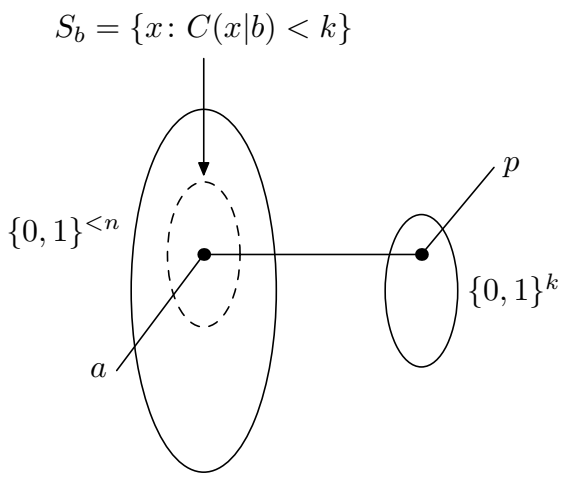


Рис. 3.4: Общая схема доказательства теоремы Мучника: слово p выбирается как сосед a в некотором графе, так чтобы внутри множество S_b мало вершин также имели p в качестве соседа.

что $|a| < n$, $\mathrm{C}(a) = n - 1$ и $\mathrm{C}(a|b) = k - 1$,[1] существует i, такое что у $f(a, i)$ имеется полиномиальное число прообразов в $S_b = \{x \mid \mathrm{C}(x|b) < k\}$. Сложность этой функции не должна превышать $O(\log n)$. Действительно, тогда $\mathrm{C}(p|a) \leqslant O(\log n)$ выполнено, т.к. $O(\log n)$ битов нужно для описания функции f и ещё $O(\log n)$ битов — для задания i. А неравенство $\mathrm{C}(a|p, b) \leqslant O(\log n)$ выполнено, поскольку $O(\log n)$ битов нужно так же для описания f и ещё $O(\log n)$ битов — для задания номера a среди прообразов p внутри S_b. Для удобства и в силу традиции в дальнейшем мы будем говорить не о функциях двух аргументов, а о двудольных графах типа (n, k, d), как в разделе 2.2. Схема доказательства проиллюстрирована на рисунке 3.4.

Разные способы доказательства отличаются друг от друга тем, как именно доказывается существование графа с необходимым свойством. Как правило, выбирается некоторое более сильное, но при этом разрешимое свойство, из которого следует нужное. При помощи вероятностного метода доказывается существование графа с этим более сильным свойством, и делается вывод о том, что один из этих графов имеет маленькую сложность. А именно, поскольку свойство разрешимо, можно в лексикографическом порядке перебрать все графы с заявленными размерами долей и степенями вершин левой доли, проверить для каждого из них выполнение свойства и взять первый, для которого свойство выполнено.

В оригинальной статье [30] Ан. Мучник в качестве определяющего свой-

[1] Здесь мы используем сделанные выше замечания про сложности и длину a.

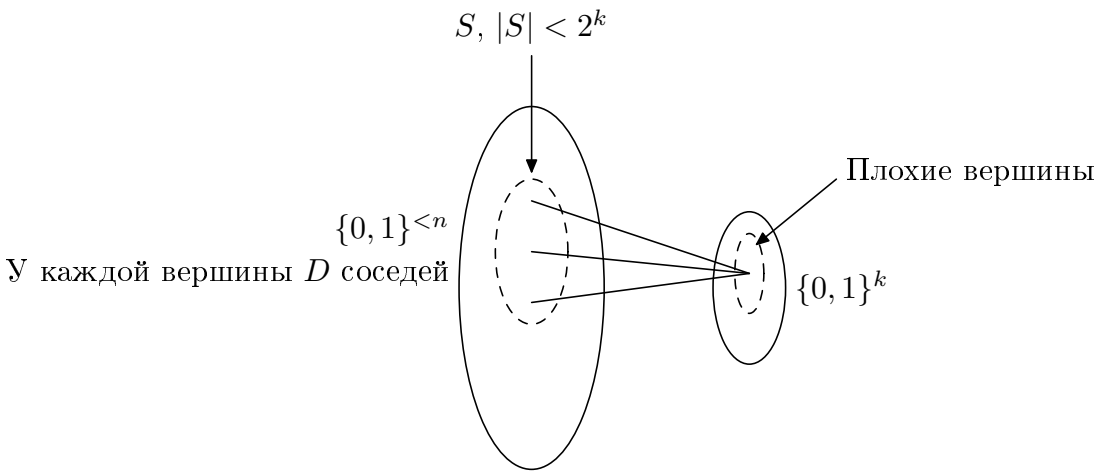


Рис. 3.5: Доказательство теоремы Мучника: плохие вершины в правой доле графа.

ства брал свойство экспандера. А. Шень использовал свойство существования онлайн-паросочетаний [42; 63; 64]. В следующем разделе мы покажем, как в качестве определяющего свойства использовать свойство экстрактора. (На самом деле, граф со свойством экстрактора можно преобразовать в граф, допускающий онлайн-паросочетания, но мы проведём прямое рассуждение).

3.2 Применение экстракторов для доказательства теоремы

В этом разделе мы вначале сформулируем промежуточное свойство, которое следует из свойства экстрактора, и из которого следует свойство, нужное в теореме Мучника, а затем докажем обе импликации.

Пусть задан двудольный граф G типа (n, m, d), а также некоторое число $k < n$ и подмножество левой доли $S \subset \{0,1\}^{<n}$ размера не больше $K = 2^k$. Назовём вершину правой доли *плохой* для S, если она имеет больше $2DK/M$ соседей внутри S, то есть хотя бы в 2 раза больше, чем в среднем (см. рис. 3.5). Назовём вершину $x \in S$ *опасной* для S, если все её соседи плохие для S (см. рис. 3.6). Наконец, назовём граф β-безопасным, если в любом S размера не больше K найдётся не больше чем βK опасных для S вершин. Безопасность графа и будет промежуточным свойством.

Следуя работе [12], докажем следующую лемму:

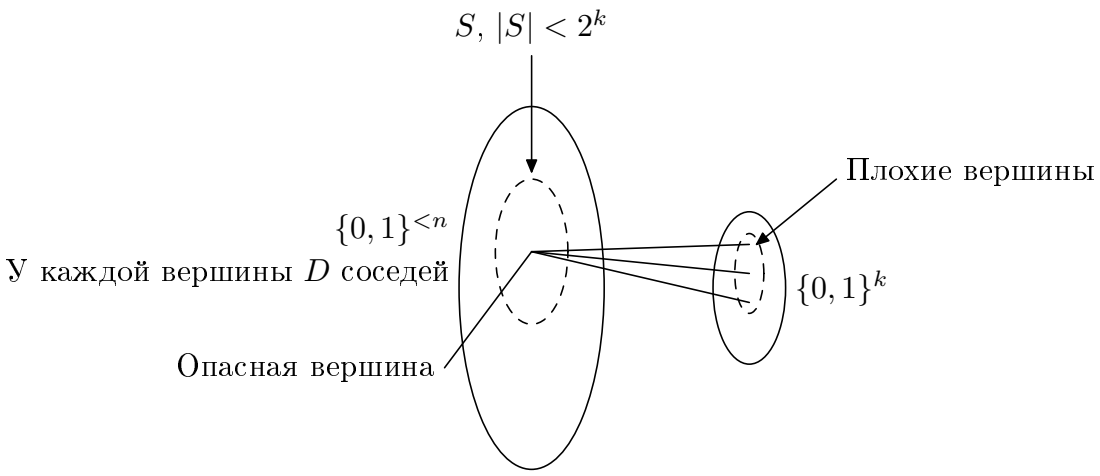


Рис. 3.6: Доказательство теоремы Мучника: опасные вершины в левой доле графа.

Лемма 3.2. *Пусть G является графом-экстрактором с параметрами n, m, d, k, ε. Тогда граф G является 2ε-безопасным.*

Доказательство. Во-первых, заметим, что без ограничения общности можно считать размер S в точности равным K: поскольку опасная вершина для подмножества является также опасной для объемлющего множества, то верхняя оценка на число опасных вершин в объемлющем множестве является верной и для подмножества.

Во-вторых, заметим, что из простых соображений (принципа Дирихле) можно вывести, что число плохих вершин не превышает половины размера правой доли. Использование свойства экстрактора позволяет уменьшить эту оценку до ε. Действительно, обозначив множество плохих вершин за Y и предположив, что $|Y| = \delta M$, мы получим, что доля плохих вершин равна $|Y|/M = \delta$, а доля рёбер, идущих в плохие вершины, т.е. $|E(S,Y)|/|E(S,R)|$, по определению плохой вершины не меньше 2δ. Далее, по определению экстрактора имеем, в частности:

$$\frac{|E(S,Y)|}{|E(S,R)|} - \frac{|Y|}{M} < \varepsilon,$$

откуда $2\delta - \delta < \varepsilon$ и $\delta < \varepsilon$, как и было заявлено.

Наконец, получим оценку на количество опасных вершин. Пусть $T \subset S$ есть множество всех опасных вершин в S. Обозначим через β отношение $\frac{|T|}{K}$. Все рёбра из T по определению ведут в множество плохих вершин Y, поэтому $|E(S,Y)| \geqslant |E(T,Y)| = |T| \cdot D = \beta DK = \beta |E(S,R)|$, откуда

$|E(S,Y)|/|E(S,R)| \geqslant \beta$. По определению экстрактора это отношение отличается от $|Y|/M = \delta$ меньше, чем на ε, откуда $\beta < \delta + \varepsilon < 2\varepsilon$. Таким образом, число опасных вершин в множестве S не превышает $2\varepsilon K$, откуда граф является 2ε-безопасным, что и требовалось. □

Доказательство теоремы Мучника. Напомним, что мы считаем, что длина a меньше n, а $\mathrm{C}(a|b) = k - 1$. Также будем считать, что $k < n$, иначе для выполнения теоремы можно взять $p = a$.

По теореме 2.19 существует экстрактор с парамерами n, k, $d = O(\log n)$, $m = k$ и $\varepsilon = 1/n^3$. Причины такого выбора ε станут ясны позднее. Один из таких экстракторов имеет сложность не больше $2\log n + O(1)$, поскольку для его описания достаточно задать n и k: остальные параметры являются функциями от этих, а свойство экстрактора является разрешимым. Все графы с параметрами (n, m, d) можно перебрать в лексикографическом порядке и проверить на свойство экстрактора. Первый полученный экстрактор будет иметь заявленную сложность. Разумеется, такой перебор требует огромных ресурсов, поэтому экстрактор не будет явным.

Зафиксируем найденный экстрактор G. Будем считать, что его левая часть индексируется словами длины меньше n (включая a),[2] а правая часть — словами длины $m = k$ (среди которых мы будем искать p). Напомним, что мы обозначаем через S_b множество $\{x \in \{0,1\}^{<n} \mid \mathrm{C}(x|b) < k\}$, и $a \in S_b$.

Поскольку $|S_b| < K$, а граф G является 2ε-безопасным, то в множестве S_b содержится не больше $2\varepsilon K$ опасных вершин. Мы докажем, что вершина a не может быть опасной, иначе она имеет слишком маленькую сложность, а раз она не опасна, то у неё есть хороший (т.е. не плохой) сосед p, который удовлетворяет всем требованиям.

Более подробно, предположим, что a не является опасной вершиной для S_b. Тогда у неё есть сосед p, который не является плохим для S_b (см. рис. 3.7). Покажем, что все требования теоремы выполнены.

Действительно, длина p равна k по построению, поэтому его сложность не превышает $k + O(1)$.

Условная сложность $\mathrm{C}(p|a)$ логарифмическая: при известном a слово p задаётся описанием графа G и номером p среди соседей a в этом графе.

[2] Существует $2^n - 1$ слов такой длины, поэтому одна вершина останется не проиндексированной. Это не повлияет на дальнейшее изложение.

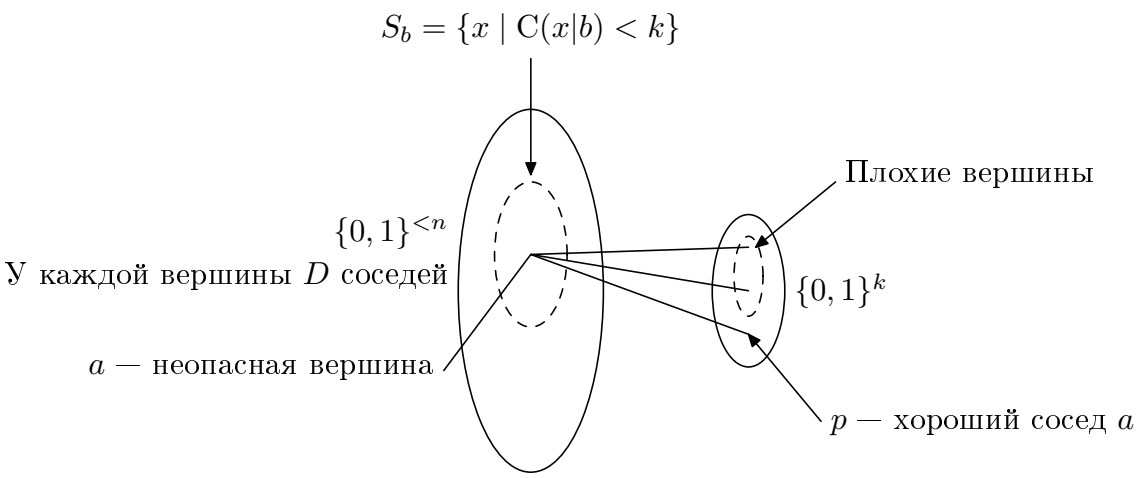


Рис. 3.7: Доказательство теоремы Мучника: выбор p.

Описание графа, как мы уже отмечали, требует $2\log n + O(1)$ битов, а запись номера p среди соседей — не больше, чем $d = O(\log n)$ битов.

Наконец, получим оценку на $\mathrm{C}(a|p,b)$. Поскольку $m = k$, а p не является плохим для S_b, то p имеет меньше $2D$ соседей в S_b. При известных n и b множество S_b можно перечислять, параллельно запуская все программы длины меньше k и выводя полученные слова длины меньше n. Если также известен граф, то можно перечислить и множество прообразов p внутри S_b. Таким образом, слово a задаётся описанием графа ($2\log n + O(1)$ битов, что также включает описание n) и номером a в последнем перечислении ($\log(2D) = O(\log n)$ битов). В сумме получается $O(\log n)$ битов, что и заявлено в теореме.

Теперь докажем, что a не может быть опасной вершиной в S_b. Напомним, что без ограничения общности мы считаем, что $\mathrm{C}(a|b) = k-1$. По лемме 3.2 опасных вершин в S_b не больше $2\varepsilon K{=}2K/n^3$. При известном графе и известном b множество опасных вершин можно перечислить: запустим перечисление S_b, и после каждого нового слова будем проверять (по определению) все уже полученные слова на опасность в данном графе для текущего приближения к S_b. Если какое-то слово оказалось опасным для приближения, то оно будет опасным и для всего S_b, поскольку множество опасных слов растёт с ростом объемлющего множества. Таким образом, при известном b каждая опасная вершина задаётся описанием графа ($2\log n + O(1)$ битов) и своим номером в перечислении ($\log(2K/n^3) = k - 3\log n + O(1)$ битов). В сумме получается $k - \log n + O(1)$ битов, таким образом a не может быть опасной вершиной, поскольку её сложность относительно b равна

$k-1$, что больше $k-\log n+O(1)$ при достаточно больших n. Заметим, что значение ε было выбрано именно так, чтобы этот вывод был справедлив.

На этом доказательство теоремы Мучника при помощи экстракторов завершено. □

3.3 Теорема Мучника для нескольких условий и префиксные экстракторы

В оригинальной статье [30] Ан.А. Мучник доказал также следующее обобщение теоремы 3.1 на случай нескольких условий:

Теорема 3.3. *Пусть даны двоичные слова a, b и c и числа n, k и l, для которых выполнено $\mathrm{C}(a) < n$, $\mathrm{C}(a|b) < k$ и $\mathrm{C}(a|c) < l$. Тогда существуют слова p и q, одно из которых является началом другого, для которых выполнены неравенства:*

- $\mathrm{C}(a|p,b) \leqslant O(\log n)$;
- $\mathrm{C}(a|q,c) \leqslant O(\log n)$;
- $\mathrm{C}(p) \leqslant k + O(\log n)$;
- $\mathrm{C}(q) \leqslant l + O(\log n)$;
- $\mathrm{C}(p|a) \leqslant O(\log n)$;
- $\mathrm{C}(q|a) \leqslant O(\log n)$.

Эта теорема ещё более нетривиальна, чем исходная. Действительно, теорема гласит, что информация об a, отсутствующая в b и c, может быть представлена двумя строками, одна из которых будет префиксом другой, даже если b и c никак не связаны. Коммуникационная схема проиллюстрирована на рисунке 3.8. Из теоремы также следует, что существует программа длины не больше $\max\{\mathrm{C}(a|b), \mathrm{C}(a|c)\} + O(\log n)$, переводящая одновременно b в a и c в a. Этот факт интересен даже без добавления о том, что такая программа может быть простой относительно a.

Особенно удивительным последнее утверждение представляется в такой ситуации: пусть b и c суть независимые (т.е. $\mathrm{dep}(b,c) = 0$) слова длины n и сложности тоже n, а слово a является их побитовой суммой: $a = b \oplus c$. Тогда $\mathrm{C}(a|b) \approx \mathrm{C}(c|b) \approx n$ (по определению независимости) и $\mathrm{C}(a|c) \approx \mathrm{C}(b|c) \approx n$. На первый взгляд кажется, что нет иного способа добиться, чтобы $\mathrm{C}(a|p,b) \leqslant O(\log n)$ и $\mathrm{C}(a|q,c) \leqslant O(\log n)$, кроме как выбрать $p = c$ и

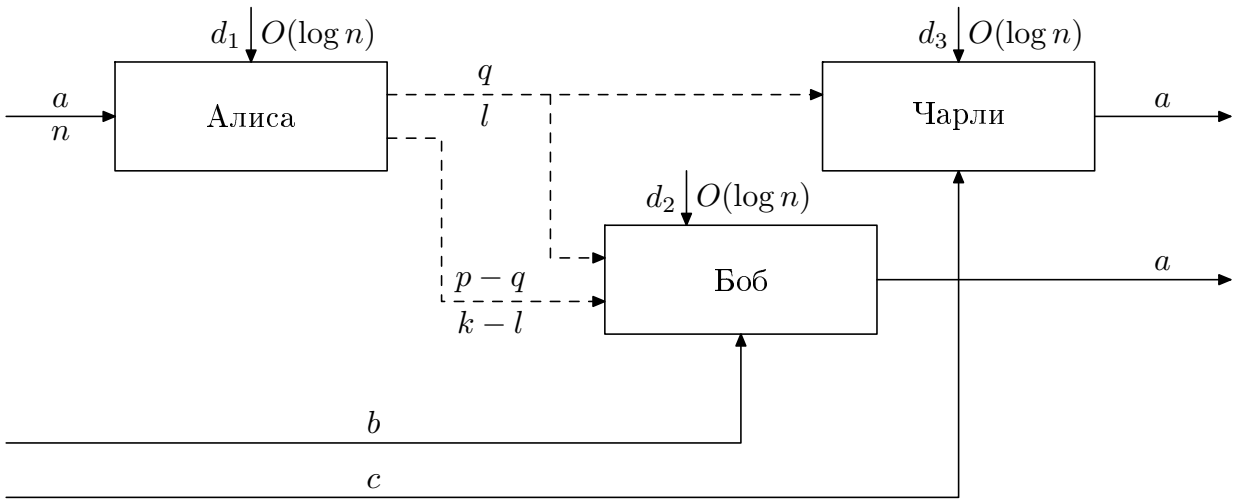


Рис. 3.8: Иллюстрация к теореме Мучника для нескольких условий. Без ограничения общности $k \geqslant l$. Алиса получает подсказку d_1 и посылает сообщение q Бобу и Чарли, а сообщение $p-q$ только Бобу. Боб, получив сообщения b, p и подсказку d_2, восстанавливает a, а Чарли по сообщениям c, q и подсказке d_3 тоже восстанавливает a.

$q = b$, или что-то очень похожее. Однако, с логарифмическими подсказками можно выбрать одно и то же слово длины примерно n в качетве p и q.

Аналогичные утверждения могут быть доказаны не только для двух, но и для нескольких, и даже для полиномиального числа условий. Мы приведём подробное доказательство теоремы для двух условий при помощи экстракторов, а затем набросаем план доказательства для многих условий. Для доказательства нам потребуется следующая модификация понятия экстрактора (соответствующее свойство неоднократно упоминалось в литературе, но термин введён только в [63]):

Определение 3.4. Будем говорить, что $(k,\, \varepsilon)$-экстрактор $\mathrm{Ext}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$, где $m \geqslant k$, является *префиксным*, если для любого $i \leqslant k$ его префикс длины $m-i$ является $(k-i,\, \varepsilon)$-экстрактором. Под префиксом понимается функция $\mathrm{Ext}_i\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{m-i}$, полученная из исходной отрезанием последних i битов.

С использованием вероятностного метода можно получить следующую теорему существования:

Теорема 3.5. *Для всех n, k и ε, для которых $1 < k \leqslant n$ и $\varepsilon > 0$, существует префиксный $(k,\, \varepsilon)$-экстрактор с параметрами $d = \log n + 2\log(1/\varepsilon) + O(1)$ и $m = k + d - 2\log(1/\varepsilon) - O(1)$.*

Доказательство. Доказательство этой теоремы похоже на стандартное доказательство теоремы 2.19. А именно, используется вероятностный метод: для случайного графа с заданными параметрами доказывается, что он обладает нужным свойством с положительной вероятностью. В отличие от стандартного доказательства, здесь нужно проверять выполнение сразу многих условий: любой префикс должен являться экстрактором. Мы покажем, что вероятность каждого из условий не просто положительна, а близка к единице, так что все условия одновременно выполнены также с положительной вероятностью.

Заметим, что в определении 2.17 достаточно потребовать истинности неравенства (2.1) только для множеств размера в точности K. Действительно, если неравенство нарушено для множества S' размера больше K и некоторого множества Y, то оно будет нарушено и для множества $S \subset S'$ точек из S', имеющих наибольшее (или наименьшее) число соседей в Y. Более того, модуль в неравенстве (2.1) можно раскрыть. Например, достаточно проверить, что

$$\frac{|E(S,Y)|}{|E(S,R)|} < \frac{|Y|}{|R|} + \varepsilon$$

для всех S размера K и всех $Y \subset R$. Действительно, неравенство

$$\frac{|E(S,Y)|}{|E(S,R)|} > \frac{|Y|}{|R|} - \varepsilon$$

следует из предыдущего, применённого к дополнению Y. Иначе говоря, если из S в Y ведёт слишком мало рёбер, то из S в дополнение Y ведёт слишком много рёбер.

Вероятностное распределение на графах определим следующим образом. Для любой вершины левой доли (слова длины n) выберем равномерно и независимо $D = 2^d$ соседей в правой части (слов длины m). Оценим сверху вероятность события *случайный граф не является префиксным экстрактором.*

Если свойство экстрактора нарушено для какого-то префикса длины $m - i$, то существуют множество $S \subset \{0.1\}^n$ из $K/2^i$ элементов и множество $Y \subset \{0,1\}^{m-i}$ размера $\alpha 2^{m-i}$ (для некоторого положительного α), количество рёбер между которыми превышает $(\alpha + \varepsilon)KD/2^i$. Здесь мы считаем, что ребро идёт в Y, если оно идёт в слово, продолжающее некоторое слово из Y. Из неравенства Хёфдинга (теорема 2.39) следует, что

вероятность этого события не превышает $\exp\left(-2\varepsilon^2 KD/2^i\right)$. Значит, вероятность события *случайный граф не является префиксным экстрактором* ограничена сверху суммой этих оценок по всем i, S и Y:

$$\sum_{i=0}^{k} C_N^{K/2^i} \cdot 2^{M/2^i} \exp\left(-2\varepsilon^2 KD/2^i\right). \tag{3.1}$$

(Здесь мы проигнорировали, что $|Y| = \alpha 2^{m-i}$, и провели суммирование по всем $Y \subset \{0,1\}^{m-i}$). В силу неравенства (2.5) выполнено $C_u^v \leqslant (ue/v)^v$, поэтому сумма (3.1) не превосходит

$$\sum_{i=0}^{k} \left(\frac{eN}{K/2^i}\right)^{K/2^i} 2^{M/2^i} \exp(-2\varepsilon^2 KD/2^i) =$$

$$= \sum_{i=0}^{k} \left(e^{K/2^i(1+\ln(2^iN/K))} \cdot e^{-\varepsilon^2 KD/2^i}\right) \cdot \left(e^{M\ln 2/2^i} \cdot e^{-\varepsilon^2 KD/2^i}\right).$$

Из условия теоремы следует, что $d = m - k + 2\log(1/\varepsilon) + O(1)$, откуда $D \geqslant \frac{M}{K} \cdot \frac{\ln 2}{\varepsilon^2}$, если константа в $O(1)$ взята достаточно большой. Значит, второй множитель в каждом слагаемом не больше 1. В то же время, первый множитель i-го слагаемого равен

$$e^{K/2^i(1+\ln(2^iN/K)-\varepsilon^2 D)} \leqslant e^{K/2^i(1+\ln N-\varepsilon^2 D)},$$

что меньше $\left(\frac{1}{2}\right)^{K/2^i}$, поскольку $d - 2\log(1/\varepsilon) = \log n + O(1)$, откуда $D\varepsilon^2 \geqslant 1 + \ln 2 + \ln N$ при достаточно большой константе в $O(1)$. Поскольку сумма этих оценок по всем i строго меньше единицы, получаем, что вероятность события *случайный граф не является префиксным экстрактором* положительна. Значит, префиксные экстракторы существуют, что и требовалось доказать. □

Однако просто использовать префиксный экстрактор недостаточно. Необходимо модифицировать рассуждение, поскольку теперь для a нужно найти два соседа в двух разных графах. Мы изменим определение опасной вершины и докажем следующий аналог леммы 3.2:

Лемма 3.6. *Назовём вершину левой доли слабо опасной в множестве S, если хотя бы половина её соседей плохи для S (см. рис. 3.9). Тогда в случае, когда граф является экстрактором, количество слабо опасных вершин в S не превышает $4\varepsilon K$.*

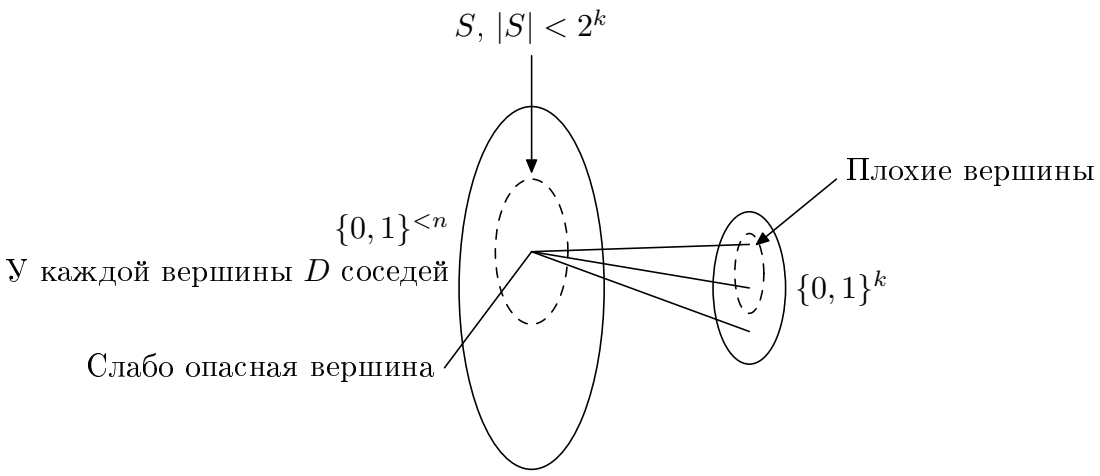


Рис. 3.9: Доказательство теоремы Мучника для двух условий: слабо опасные вершины в левой доле графа.

Доказательство. Доказательство почти дословно повторяет доказательство леммы 3.2. Отличие состоит в том, что теперь лишь половина рёбер из множества опасных вершин T ведёт в множество плохих вершин Y, откуда $|E(T,Y)| \geqslant \frac{1}{2}|T| \cdot D = \frac{1}{2}\beta|E(S,R)|$ и $|E(S,Y)|/|E(S,R)| \geqslant \frac{1}{2}\beta$. Далее аналогично доказательству леммы 3.2 получаем $\frac{1}{2}\beta < 2\varepsilon$, откуда $|T| < 4\varepsilon K$, что и требовалось. □

Теперь дадим новое доказательство теоремы 3.3, основанное на использовании префиксных экстракторов. Зафиксируем префиксный экстрактор E с параметрами n, k, $d = O(\log n)$, $m = k$ и $\varepsilon = 1/n^3$. Как и прежде, мы можем считать сложность этого экстрактора равной $2\log n + O(1)$. Также можно без ограничения общности предположить, что $|a| < n$, $\mathrm{C}(a|b) = k - 1$, $\mathrm{C}(a|c) = l - 1$ и $k \geqslant l$.

Обозначим через S_b и S_c множества слов длины меньше n и сложности меньше k и l условно на b и c соответственно. Называя слово слабо опасным в S_b, будем иметь в виду слабую опасность в исходном графе, а называя его слабо опасным в S_c, — слабую опасность в графе, соответствующем l-битовому префиксу. Поскольку этот префикс E_{k-l} также является экстрактором, лемма 3.6 верна и для него. Слово a лежит в пересечении S_b и S_c и не является слабо опасным ни в одном из множеств, иначе сложность $\mathrm{C}(a|b)$ или сложность $\mathrm{C}(a|c)$ была бы меньше заявленной. (Здесь рассуждение полностью аналогично рассуждению при доказательстве исходной теоермы Мучника). Значит, меньше половины соседей a в графе E плохи

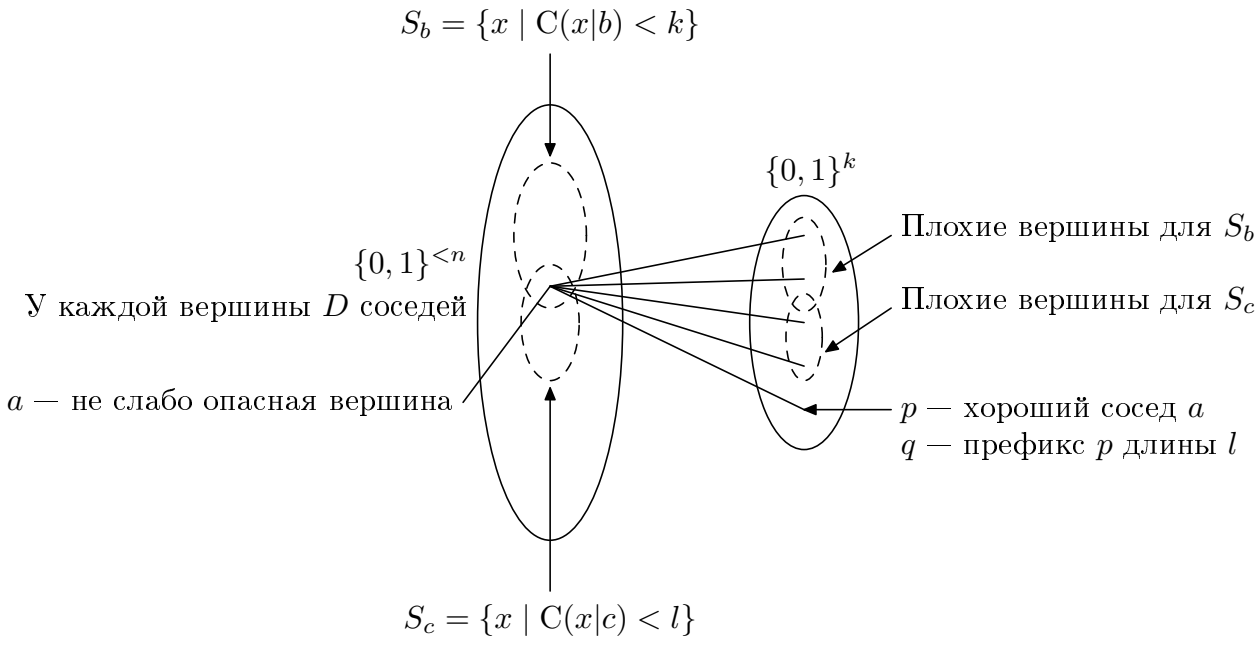


Рис. 3.10: Доказательство теоремы Мучника: выбор p.

для S_b и меньше половины соседей a в графе E_{k-l} плохи для S_c. Последнее означает, что меньше половины соседей a в графе E имеют l-битные префиксы, которые плохи для S_c. Таким образом, найдётся такой сосед p, что он сам не будет плохим для S_b, а его l-битный префикс q не будет плохим для S_c. Эти p и q будут искомыми (см. рис. 3.10).

Действительно, условные сложности $\mathrm{C}(p|a)$ и $\mathrm{C}(q|a)$ логарифмические, потому что p и q можно задать экстрактором и номерами этих слов среди соседей a. Длины p и q соответствуют требованиям. Наконец, слово a можно получить из p и b, задав экстрактор E_k и номер a среди соседей p в множестве S_b. Поскольку p не будет плохим, размер описания будет логарифмическим. Аналогично, a можно получить из q и c, задав экстрактор E_{k-l} и номер a среди соседей q в множестве S_c, размер описания будет также логарифмическим.

Таким образом, все условия теоремы Мучника для двух условий проверены, а сама теорема доказана. □

Сформулируем теперь теорему Мучника для многих условий.

Теорема 3.7. *Пусть даны числа n, $s = \mathrm{poly}(n)$ и $k_1 \leqslant \ldots \leqslant k_s$ и двоичные слова a, b_1, …, b_s, такие что $\mathrm{C}(a) < n$ и $\mathrm{C}(a|b_i) < k_i$ при всех $i = 1, \ldots, s$. Тогда существуют слова p_1, …, p_s, для которых выполнены следующие условия:*

- p_i *является началом* p_j *при* $i < j$*;*
- $\mathrm{C}(a|p_i, b_i) \leqslant O(\log n)$ *при всех* $i = 1, \ldots, s$*;*
- $\mathrm{C}(p_i) \leqslant k_i + O(\log n)$ *при всех* $i = 1, \ldots, s$*;*
- $\mathrm{C}(p_i|a) \leqslant O(\log n)$ *при всех* $i = 1, \ldots, s$ *(достаточно* $\mathrm{C}(p_s|a) \leqslant O(\log n)$*).*

Доказательство этой теоремы обобщает доказательство теоремы 3.3, не привнося в него никаких новых идей, поэтому мы лишь кратко опишем его. Вначале нужно взять префиксный экстрактор с $m = k_s$, $\varepsilon < \frac{1}{sn^3}$ и $d = \log s + O(\log n) = O(\log n)$. Такой экстрактор существует по теореме 3.5. Затем нужно доказать аналог леммы 3.6, определив слабо опасную вершину как ту, у которой хотя бы доля $1/s$ соседей (в соответствующем экстракторе) плохи для S. Количество слабо опасных вершин тогда не превысит $2s\varepsilon K = 2K/n^3$. Наконец, вершина a вновь не будет слабо опасной ни в одном из множеств S_{b_i}, откуда следует существование у a соседа, хорошего для всех множеств S_{b_i}. Этот сосед и будет искомым p_s, а все остальные p_i — его началами соответствующей длины.

Глава 4

Теорема Мучника для сложности с ограничением на память

В этой главе мы докажем несколько вариаций теоремы Мучника для колмогоровской сложности с ограничением на используемую память. Для доказательства первой вариации можно использовать любой явный экстрактор. Формулировка и доказательство соответствующей теоремы приводятся в разделе 4.1. В доказательстве другой вариации мы применим идею дерандомизации: заменим случайный граф, использованный в доказательстве теоремы Мучника, на псевдослучайный, полученный генератором Нисана–Вигдерсона. При этом свойство экстрактора уже не понадобится: в доказательстве будет использовано более простое свойство. В разделе 4.2 мы сформулируем это свойство, докажем его соблюдение для псевдослучайного графа и подробно опишем процесс дерандомизации. Наконец, в разделе 4.3 мы изложим обе вариации для теоремы с несколькими условиями.

Поскольку в этой главе идёт речь только о сложности с ограничением на память, верхний индекс в обозначениях сложности будет всегда обозначать ограничение на используемую память.

4.1 Доказательство при помощи явного экстрактора

Докажем теорему Мучника для сложности с ограничением на память в следующей формулировке:

Теорема 4.1. *Пусть даны двоичные слова a и b, а также числа n, k и s, для которых верно $\mathrm{C}^s(a) < n$ и $\mathrm{C}^s(a|b) < k$. Пусть числа d и q таковы, что при любом $l \leqslant k$ существует $(l,\ 0.25)$-экстрактор*

$\mathrm{Ext}_l\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^l$, *вычислимый на памяти* q *(для некоторой заранее фиксированной универсальной многоленточной машины Тьюринга). Тогда существует такое слово* p*, что:*

- $\mathrm{C}^{O(s+q+n^2)}(a|p,b) \leqslant d + O(\log n)$*;*
- $|p| \leqslant k + O(\log n)$*;*
- $\mathrm{C}^{O(q)}(p|a) \leqslant d + O(\log n)$*.*

Доказательство. Основные шаги повторяют доказательство теоремы Мучника, представленное в разделе 3.2, однако потребуются и дополнительные рассуждения. Сначала мы докажем ослабленную версию теоремы, в которой число s добавлено в сложность из первого неравенства в качестве условия: $\mathrm{C}^{O(s+q+n^2)}(a|p,b,s) \leqslant d + O(\log n)$. Затем мы покажем, как избавиться от этого дополнительного условия.

Рассмотрим экстрактор Ext_k, существующий по условию теоремы. Напомним, что элемент его правой доли называется плохим для подмножества S левой доли, если он имеет больше, чем $2DK/M$ соседей в S. (Размер S не больше K). Элемент $x \in S$ называется опасным в S, если все его соседи плохие для S. Как и в исходном доказательстве, разберём два случая: когда a является или не является опасным в множестве $S_{b,s} = \{x \mid \mathrm{C}^s(x|b) < k\}$. В отличие от исходного доказательства, опасность a не приводит к противоречию, а требует дополнительной конструкции.

Первый случай. Слово a не является опасным. Тогда у него есть сосед p, не являющийся плохим. Покажем, что он искомый. Действительно, его длина соответствует условию по построению. Далее, для его описания при известном a нужно задать n и k, достаточные для описания Ext_k, и его номер среди всех соседей a в этом графе. Для первого нужно не больше $3\log n$ битов, для второго — d битов. Поскольку экстрактор вычислим на памяти q, то и вычисление p при помощи Ext_k потребует такой памяти. За счёт перехода к оптимальному способу описания память вырастет не более, чем в константу раз, что и даст нужную в третьем неравенстве оценку.

Наконец, покажем выполнение первого неравенства. Как и прежде, при известных b и s можно перечислять множество $S_{b,s}$, запуская оптимальный способ описания на всех программах длины не больше k (и b в качестве второго аргумента) и ограничивая используемую им память величиной s. Как только очередное слово получено, нужно проверить, не является ли p одним из соседей этого слова в экстракторе Ext_k. Поскольку p хорошее, у него

не больше $2D$ соседей в $S_{b,s}$, поэтому a можно задать номером в описанном переборе. Таким образом, помимо параметров экстрактора, занимающих $3\log n$ битов, потребуется $d + O(1)$ битов информации, что соответствует заявленной оценке. Проверим, что выполнены требования по занимаемой памяти. Действительно, для симуляции вычислений на памяти s требуется $O(s + n)$ ячеек: дополнительная память нужна для контроля невыхода за пределы зоны s ($\log s$ ячеек), для контроля отсутствия зацикливания (s ячеек: нужно считать количество шагов и контролировать, что оно не превышает 2^s, превышение говорит о зацикливании) и хранения промежуточных результатов ($O(n)$ ячеек). Также нужна память q для вычисления значения Ext_k. При этом вычисления, занимающие s ячеек, проводятся последовательно, поэтому можно использовать одну и ту же зону. То же верно для вычислений, занимающих память q. Переход к оптимальному способу описания может увеличить использованную память в константное число раз, откуда и получается общая оценка $O(s + q + n)$.[1]

Второй случай. Слово a является опасным. Вначале попробуем провести рассуждение аналогично основной теореме и покажем, почему получить противоречие аналогичным образом не получится. Заметим, что плохие для $S_{b,s}$ слова можно перечислять на памяти $O(s + q + n)$, получив на вход n, k, b и s. Действительно, для сложности с ограничением на память перечисление эквивалентно распознаванию, а распознать, является ли y плохим, можно, перечисляя элементы $S_{b,s}$, затем запуская Ext_k на полученных словах со всеми возможными вторыми аргументами и считая, сколько раз y встретилось среди образов. Оценка на использованную память проводится аналогично первому случаю.

Далее, опасные в $S_{b,s}$ элементы также можно перечислять на памяти $O(s + q + n)$. Действительно, такой памяти хватит для перечисления самого $S_{b,s}$, вычисления всех соседей каждого слова в Ext_k и выяснения про каждого соседа, является ли он плохим. Поскольку опасных слов не больше $2\varepsilon K$, каждое из них имеет сложность (условно на n, k, b, s) не больше $k + \log\varepsilon + O(1)$. К сожалению, противоречия тут не получится, даже если взять $\varepsilon = 1/n^3$, как в исходной теореме, вместо 0.25 в текущей. Ведь мы доказали лишь то, что опасные слова имеют меньшую сложность при более мягком ограничении на использованную память, а здесь нет противоречия.

[1] Возможно, более естественно написать $O(\max\{s, q, n\})$, что даёт ту же оценку с точностью до константы в $O(\cdot)$-обозначении

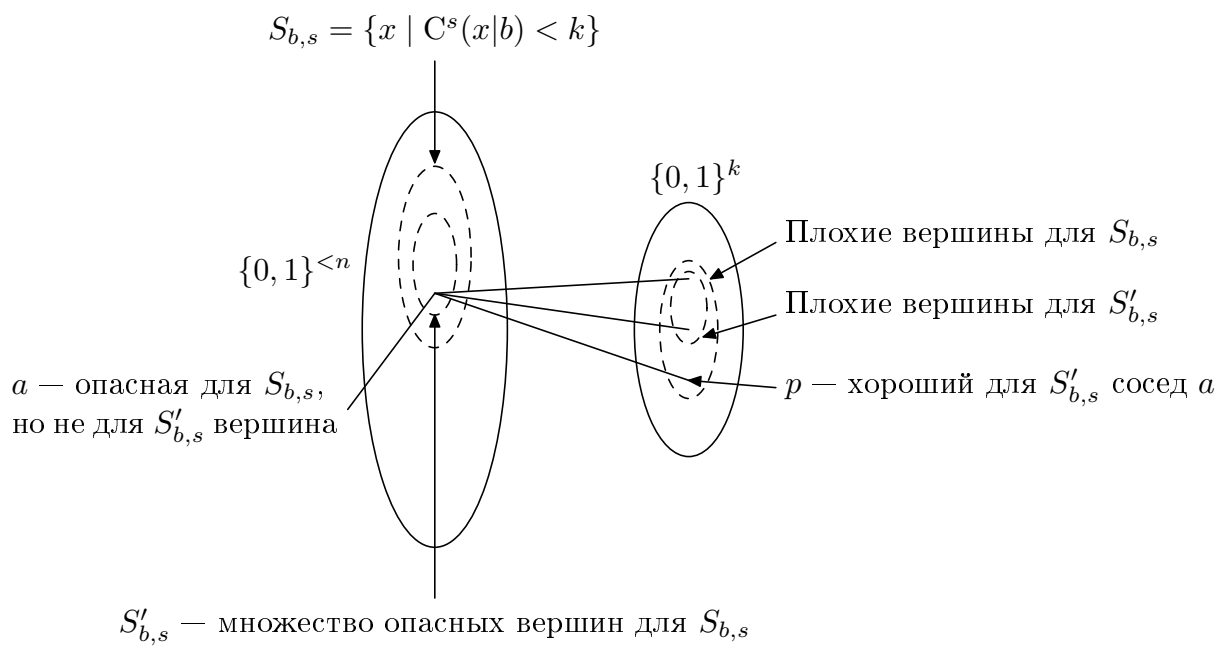


Рис. 4.1: Доказательство теоремы Мучника с ограничением на память: выбор p для опасного a.

Вместо этого мы воспользуемся тем, что опасных слов немного, и построим для них отдельную итеративную конструкцию. Мы вначале подробно опишем первые шаги, а потом — всю конструкцию. Обозначим через $S'_{b,s}$ множество опасных слов для $S_{b,s}$. Его размер не больше $2\varepsilon K = K/2$. Введём обозначения $K_1 = 2\varepsilon K$ и $k_1 = \log K_1$. Заметим, что $k_1 < k$, и рассмотрим экстрактор Ext_{k_1}, существующий по условию теоремы. Далее, рассмотрим «опасные слова второго порядка», т.е. опасные слова в экстракторе Ext_{k_1} в множестве $S'_{b,s}$. По лемме 3.2 опасных слов второго порядка не больше $2\varepsilon K_1$. Если слово a (лежащее в $S'_{b,s}$) не является опасным второго порядка, то у него есть хороший сосед в Ext_{k_1}, который можно взять в качестве искомого p. (Доказательство полностью повторяет первый случай). Если же слово a является опасным второго порядка, то нужно применить ещё одну итерацию. Обозначим через $S''_{b,s}$ множество опасных слов второго порядка. Его размер не больше $K_2 = 2\varepsilon K_1 = K_1/2$, поэтому $k_2 = \log K_2 < k_1$. Рассмотрим экстрактор Ext_{k_2}, существующий по условию теоремы. В нём есть «опасные слова третьего порядка», т.е. опасные слова в экстракторе Ext_{k_2} в множестве $S''_{b,s}$. По лемме 3.2 опасных слов третьего порядка не больше $2\varepsilon K_2$. Если слово a не является опасным третьего порядка, то у него есть хороший сосед в Ext_{k_2}, который можно взять в качестве искомого. Иначе нужно сделать ещё одну итерацию, и так далее.

Выбор для первой итерации проиллюстрирован на рис. 4.1.

Опишем теперь всю конструкцию целиком. Пусть $k_i = k - i$, а $K_i = 2^{k_i} = K/2^i$, где $i = 0, 1, \ldots, k$. Обозначим через $S^{(0)}_{b,s}$ множество $S_{b,s}$, а через $S^{(i+1)}_{b,s}$ — множество опасных слов в множестве $S^{(i)}_{b,s}$ в экстракторе Ext_{k_i}, существующем по условию теоремы. Применяя индуктивно лемму 3.2, получаем, что множество $S^{(i)}_{b,s}$ содержит меньше 2^{k_i} элементов. Поскольку $k_k = 0$, множество $S^{(k)}_{b,s}$ пусто. Значит, для некоторого i слово a лежит в $S^{(i)}_{b,s}$, но не лежит в $S^{(i+1)}_{b,s}$. Значит, в экстракторе Ext_{k_i} у a есть хороший сосед p. Докажем, что он удовлетворяет всем условиям теоремы.

Действительно, длина слова p удовлетворяет условию по построению. Сложность p при известном a мала: нужно задать n и k_i для построения экстрактора ($O(\log n)$ битов), а также номер p среди соседей a (d битов), в сумме $d + O(\log n)$ битов. Наконец, сложность a при известных b, p и s также мала: нужно задать n, k_i и номер a среди соседей p в множестве $S^{(i)}_{b,s}$. Поскольку $a \notin S^{(i+1)}_{b,s}$, последний номер требует не больше $d + O(1)$ битов, что даёт заявленную оценку. Однако для получения оценки на использованную память нужно доказать, что множество $S^{(i)}_{b,s}$ можно перечислить на памяти $O(s + q + n^2)$.

Докажем по индукции, что верно более сильное утверждение: множество $S^{(i)}_{b,s}$ можно перечислить на памяти $O(s + q + n(i + 1))$. База (для $i = 0$) была доказана при рассмотрении первого случая. Там же, в начале рассмотрения второго утверждения мы фактически доказали следующее: если множество S перечислимо на памяти $r = O(s + q + n)$, то и множество опасных элементов в S в данном экстракторе перечислимо на памяти $r + O(n)$. Из этого утверждения непосредственно следует переход, поскольку $S^{(i+1)}_{b,s}$ есть множество опасных элементов множества $S^{(i)}_{b,s}$ в экстракторе Ext_{k_i}. Таким образом, индуктивное утверждение доказано, значит множество $S^{(i)}_{b,s}$ действительно можно перечислить на памяти $O(s + q + n^2)$, и теорема в ослабленной формулировке доказана.

Избавление от дополнительного условия теоремы. Теперь покажем, как доказать теорему в исходной формулировке, т.е. без добавления s в первую условную сложность. Знание s использовалось при перечислении множества $S_{b,s}$ для ограничения зоны работы программ. Хотя при неизвестном s перечислить $S_{b,s}$ на памяти $O(s+n)$ в общем случае невозможно, мы покажем, как обойтись без такого перечисления. Заметим, что S_b является

объединением множеств $S_{b,s}$ по всем $s = 1, 2, 3, \ldots$. Поэтому множество S_b можно перечислять, перечисляя последовательно $S_{b,1}, S_{b,2}, S_{b,3}$, и так далее. Можно добиться, чтобы каждое слово из S_b встретилось в этом перечислении ровно один раз. Это нужно делать так: запустить перечисление $S_{b,s}$, запуская в лексикографическом порядке все программы длины не больше k на входе b на памяти s с контролем зацикливания. Как только некоторая программа z выдала результат x, нужно проверить, что он не встречался ранее. Для этого, во-первых, нужно запустить перечислитель $S_{b,s-1}$ (т.е. запустить все программы длины не больше k на входе b на памяти $s-1$) и проверить, не встречается ли там x. Во-вторых, нужно запустить все программы от 0 до $z-1$ на памяти s и проверить, что ни одна из них не выдаёт x. Только если обе проверки пройдены, нужно добавить x в перечисление. Приведём также немного оптимизированный псевдокод:

Вход: Слово b, число k
Результат: Перечисление $S_b = \{x \mid \mathrm{C}(x|b) < k\}$ без повторений

```
1  для s = 1 до ∞ выполнить
2      для каждого z ∈ {0,1}^{<k} выполнить
3          Запустить U(z, b) на зоне s с контролем зацикливания;
4          если U(z, b) корректно завершилось то
5              x := U(z, b);
6              для каждого w ∈ {0,1}^{<k} выполнить
7                  если w < z то q := s;
8                  иначе q := s − 1;
9                  Запустить U(w, b) на зоне q с контролем зацикливания;
10                 если U(w, b) корректно завершилось и вернуло x то
11                     продолжить цикл по z;
12                 конец условия
13             конец цикла
14             вывести x ;          /* Выполняется, только если прошли
               проверки для всех w */
15         конец условия
16     конец цикла
17 конец цикла
```

Алгоритм 4.1: Перечисление S_b в нужном порядке без повторений

Можно построить это перечисление таким образом, что к моменту, когда перечислены все элементы $S_{b,s}$, использована только память $O(s+n)$: для

этого нужно каждый раз запускать очередную программу на одной и той же памяти, храня между запусками только x, z и счётчики для проверки того, что x не встречалось ранее. Разумеется, для новой проверки нужно также использовать старый счётчик. Таким образом, к моменту перечисления всего $S_{b,s}$ будет использовано $O(s)$ ячеек рабочей памяти и $O(n)$ ячеек для хранения счётчиков.[2]

Более того, можно аналогичным образом перечислить множества $S_b^{(i)}$ для всех i. (Здесь $S_b^{(i)}$ — множество всех опасных слов для $S_b^{(i-1)}$ в экстракторе $\mathrm{Ext}_{k_{i-1}}$, а $S_b^{(0)} = S_b$). Действительно, поскольку множество $S_{b,s}^{(i)}$ перечислимо на памяти $O(s+n)$ при всех i, можно запустить полностью аналогичную конструкцию. Приведём соответствующий псевдокод:

Вход: Слово b, числа k, i
Результат: Перечисление $S_b^{(i)}$ без повторений

```
если i=0 то
    Запустить алгоритм 4.1
иначе
    для каждого z из перечисления S_b^{(i-1)} выполнить
        если z опасно в экстракторе Ext_{k_{i-1}} то вывести z;
    конец цикла
конец условия
```

Алгоритм 4.2: Перечисление $S_b^{(i)}$ в нужном порядке без повторений

Если перечисление построено именно таким образом, то нет нужды знать s, чтобы восстановить a по b и p. Действительно, a просто задаётся своим номером в перечислении соседей p внутри S_b. К тому моменту, когда a будет найдено, перечисление (части) S_b использует память $O(s+n)$. Добавляя пространство для вычисления значений экстрактора (чтобы оставить в перечислении только соседей p) и хранения промежуточных результатов, получим заявленную оценку $O(s+q+n^2)$. Таким образом, теорема в исходной формулировке доказана. □

Заметим, что для опасных слов длина программы p получилась гарантированно меньшей $\mathrm{C}^s(a|b)$. Это явление в некотором роде аналогично тому, что без ограничения на память a не могло быть опасным.

Завершим раздел следствием для известной конструкции экстракторов.

[2]Заметим, что хранить все уже перечисленные слова и программы, которыми они получены, невозможно, поэтому приходится заново их генерировать.

Следствие 4.2. *Пусть даны двоичные слова a и b, а также числа n, k и s, для которых верно $\mathrm{C}^s(a) < n$ и $\mathrm{C}^s(a|b) < k$. Тогда существует такое слово p, что:*

- $\mathrm{C}^{O(s+\mathrm{poly}(n))}(a|p,b) \leqslant O(\log^3 n)$;
- $|p| \leqslant k + O(\log n)$;
- $\mathrm{C}^{\mathrm{poly}(n)}(p|a) \leqslant O(\log^3 n)$.

Доказательство. Утверждение получается непосредственной подстановкой экстрактора из следствия 2.22 в теорему 4.1. □

Замечание 4.3. Как и для теоремы без ресурсных ограничений, можно добиться, чтобы p было длины ровно k, либо чтобы $\mathrm{C}^{O(s+\mathrm{poly}(n))}(a|p,b) \leqslant O(1)$, но при этом $|p| \leqslant k + O(\log^3 n)$. В обоих случаях растёт $\mathrm{C}^{\mathrm{poly}(n)}(p|a)$, оставаясь порядка $\log^3 n$.

4.2 Доказательство при помощи генератора Нисана–Вигдерсона

Доказательство теоремы Мучника для сложности без ограничений использовало вероятностный метод: было доказано, что случайный двудольный граф обладает определённым свойством (а именно, является экстрактором, ещё точнее — безопасным графом) с положительной вероятностью. Это соображение нельзя использовать для доказательства теоремы с полиномиальным ограничением на память, поскольку перебор всех графов требует экспоненциальной памяти. В разделе 4.1 мы показали, как использовать для доказательства теоремы явную конструкцию. В этом разделе мы покажем, как дерандомизировать исходную конструкцию, заменив случайный граф на псевдослучайный. При этом все невязки в условиях теоремы вновь примут вид $O(\log n)$.

Идея состоит в том, чтобы использовать генератор псевдослучайных чисел Нисана–Вигдерсона, существующий по следствию 2.36. Для того, чтобы применить лемму 2.37, нужно построить схему константной глубины, распознающую свойство, которое можно использовать для вывода теоремы Мучника. В работе [62] автором было использовано довольно сложное свойство, следующее из свойства экстрактора по лемме 3.2. Было показано, что это свойство можно распознать схемой константной глубины. Однако,

впоследствии благодаря совету Ронена Шалтиеля было получено гораздо более простое свойство, которое столь же успешно можно использовать в доказательстве. Более того, двудольные графы (т.е. наборы функций) больше не понадобятся, достаточно будет использовать одну функцию из $\{0,1\}^n$ в $\{0,1\}^k$.

План дальнейшего изложения таков: вначале мы сформулируем это свойство, затем докажем, что случайная функция обладает этим свойством с отделённой от нуля вероятностью, затем докажем, что это свойство можно распознать схемой из функциональных элементов константной глубины. Из этого будет следовать, что наше свойство с положительной вероятностью встречается среди образов генератора Нисана–Вигдерсона. Затем мы покажем, что аргумент генератора, являющийся прообразом функции с этим свойством, можно найти алгоритмически, используя небольшое количество памяти. Наконец, мы сформулируем и докажем вариант теоремы Мучника, который можно получить из этой конструкции.

4.2.1 Формулировка нужного свойства и доказательство существования

Пусть дана функция $F\colon \{0,1\}^n \to \{0,1\}^k$. Пусть $\mathcal{S}$ — система подмножеств $\{0,1\}^n$, состоящая из не более чем 2^{n+k} подмножеств, каждое из которых имеет размер не больше 2^k. Скажем, что F имеет меньше m коллизий относительно $\mathcal{S}$, если для любого $x \in \{0,1\}^k$ и для любого $S \in \mathcal{S}$ элемент x имеет менее m прообразов внутри S.

Заметим, что верхнее ограничение на размер системы $\mathcal{S}$ не тривиально: общее количество подмножеств $\{0,1\}^n$ размера не больше 2^k имеет порядок больше 2^{2^k}, что больше 2^{n+k}.

Теорема 4.4. *Существует полином $p(n)$, такой что для любых $n > 1$ и $k \leqslant n$ и любой системы $\mathcal{S}$, состоящей из не более чем 2^{n+k} подмножеств $\{0,1\}^n$, каждое из которых имеет размер не больше 2^k, хотя бы половина всех функций $F\colon \{0,1\}^n \to \{0,1\}^k$ имеют меньше $p(n)$ коллизий относительно $\mathcal{S}$.*

Доказательство. Для доказательства используем вероятностный метод. Рассмотрим случайную функцию F, значения которой выбраны равномерно и независимо. Оценим сверху число α — вероятность того, что F имеет

не меньше $p(n)$ коллизий относительно $\mathcal{S}$, и покажем, что эта оценка меньше одной второй при правильном выборе полинома $p(n)$. Из такой оценки сразу следует утверждение теоремы.

Если F имеет не меньше m коллизий, то хотя бы для одного элемента $x \in \{0,1\}^k$ найдётся хотя бы m прообразов этого элемента внутри хотя бы одного множества $S \in \mathcal{S}$. Поскольку вероятность объединения событий не превосходит суммы вероятностей, α не превосходит $2^{n+k} \cdot 2^k \cdot C_{2^k}^m \cdot \left(\frac{1}{2^k}\right)^m$. Здесь 2^{n+k} — количество всевозможных S, 2^k — количество всевозможных x, $C_{2^k}^m$ — оценка сверху количества всевозможных m-элементных подмножеств множества S, $\left(\frac{1}{2^k}\right)^m$ — вероятность того, что образы всех m элементов равны x. Поскольку $C_{2^k}^m < \frac{\left(2^k\right)^m}{m!}$, получаем, что $\alpha < \frac{2^{n+2k}}{m!} \leqslant \frac{2^{3n}}{m!}$. Если взять $m = 3n$, то получим $\alpha < 1/2$ при всех $n > 1$ и $k \leqslant n$. Из полученной оценки следует утверждение теоремы. □

4.2.2 Распознавание малого числа коллизий при помощи схемы константной глубины

Функцию из $\{0,1\}^n$ в $\{0,1\}^k$ можно описать словом из нулей и единиц длины $k2^n$: для каждого из 2^n аргументов нужно указать одно значение длины k. Таким образом, осмысленен вопрос о том, можно ли то или иное свойство такой функции распознать некоторой схемой из функциональных элементов со входом длины $k2^n$ и одним выходом. Мы покажем, что малость числа коллизий можно распознать схемой из функциональных элементов константной глубины. Формально мы докажем следующую теорему:

Теорема 4.5. *Существует константа d, такая что для всех $k \leqslant n$ и любого семейства $\mathcal{S}$, состоящего из не более чем 2^{n+k} подмножеств размера не больше 2^k, существует схема из функциональных элементов $C_{\mathcal{S}}$ размера $2^{O(n)}$ и глубины d, получающая на вход код функции $F\colon \{0,1\}^n \to \{0,1\}^k$ и выдающая 1 тогда и только тогда, когда функция F имеет не больше $3n$ коллизий относительно $\mathcal{S}$.*

Доказательство. Описание функции F будем подавать на вход схеме как список значений F на всех входах. (Входы будут упорядочены, например, лексикографически). Схема будет представлять собой конъюнкцию похожих схем, каждая из которых будет проверять, что коллизий для отдельного S мало. Для каждого S на вход соответствующей копии будут поданы

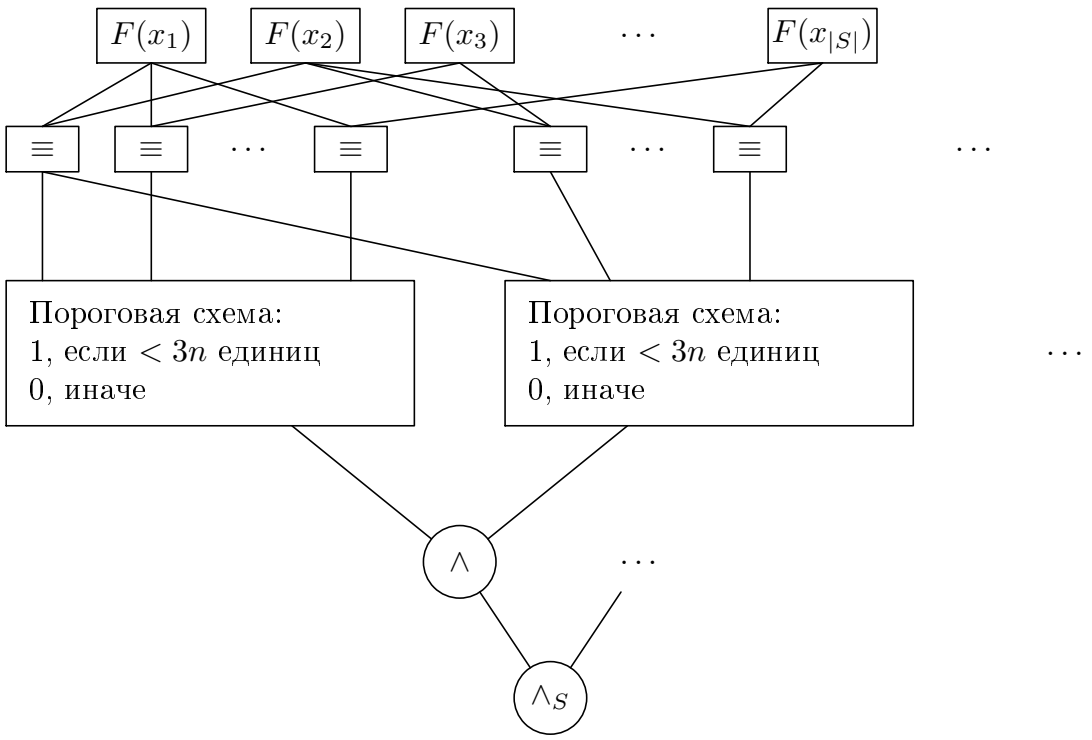


Рис. 4.2: Эскиз схемы, проверяющей малое число коллизий у хеш-функции.

значения F на всех элементах из S. Схема должна распознать, найдутся ли среди них $3n+1$ одинаковых.

Это делается так: вначале для каждой пары значений вычисляется, совпадают они или нет. Это делается совсем простой схемой: берутся эквиваленции от соответствующих друг другу битов, а затем конъюнкция результатов.

Затем для каждого элемента $x \in S$ берутся результаты сравнения $F(x)$ со всеми остальными значениями $F(y)$, $y \in S$. К ним применяется пороговая схема, существующая по теореме 2.33: она выдаёт 1, если среди входов меньше $3n$ единиц, и 0 в противном случае.[3]

Наконец, нужно взять конъюнкцию по всем $x \in S$, а затем по всем $S \in \mathcal{S}$. Построенная схема (см. рис. 4.2) удовлетворяет всем условиям. Действительно, она имеет константную глубину как композиция трёх схем константной глубины. Она имеет размер $2^{O(n)}$: схема для конкретного S имеет размер $O(k2^{2k}) + 2^k \cdot \mathrm{poly}(2^k) = 2^{O(n)}$, а всего различных S не больше $2^{n+k} = 2^{O(n)}$ штук, в произведении всё равно получится $2^{O(n)}$. Формула для размера схемы для одного S объясняется так: на первом этапе нужно

[3]Если k существенно меньше n, так что $3n$ не будет полилогарифмической функцией от 2^k, то в схему нужно добавить недостающее число фиктивных битов входа, равных нулю.

сравнить $\frac{|S|(|S|-1)}{2} = O(2^{2k})$ пар значений функции F, на одно сравнение нужно $O(k)$ элементов. На втором шаге нужно для каждого из 2^k слов x применить схему из теоремы 2.33. Эта схема имеет полиномиальный размер от числа своих входов, т.е. её размер есть $\mathrm{poly}(2^k) = 2^{O(k)}$. Конъюнкция добавляет ещё один элемент.

Наконец, построенная схема возвращает заявленный результат. Если F имеет не больше $3n$ коллизий, то для любого S и любого $x \in S$ значение $F(x)$ совпадает менее, чем с $3n$ значениями $F(y)$ для других $y \in S$. В таком случае всех схемы второго уровня возвратят 1, и конъюнкция всех этих значений также будет равна 1. Если же в каком-то S есть $3n+1$ коллизия, то соответствующая схема возвратит 0, а значит, и конъюнкция возвратит 0. □

Теперь, применяя базовый принцип 2.37, теорему существования 4.4 и теорему о распознающей схеме 4.5, можно заключить следующее:

Следствие 4.6. *Пусть $G_n\colon \{0,1\}^l \to \{0,1\}^N$ — генератор Нисана–Вигдерсона, где $N = k2^n$, $l = O(n^{2d+6})$, а d — глубина схемы, построенной в теореме 4.4. Тогда для любого семейства $\mathcal{S}$ из не более 2^{n+k} подмножеств $\{0,1\}^n$ размера не больше 2^k среди образов $G_n(x)$ существует код функции $F\colon \{0,1\}^n \to \{0,1\}^k$, имеющей не больше $3n$ коллизий.*

4.2.3 Поиск аргумента генератора Нисана–Вигдерсона, порождающего функцию с малым числом коллизий

До сих пор изложение не опиралось на конкретный вид системы множеств $\mathcal{S}$. В этом разделе мы опишем, какую систему нужно рассмотреть для доказательства варианта теоремы Мучника, а также докажем, что аргумент x, образ которого $G_n(x)$ описывает функцию с малым числом коллизий, можно найти алгоритмически, использовав небольшую память.

Утверждение 4.7. *Зафиксируем некоторое число $\overline{q}$. Пусть $\mathcal{S} = \big\{S \mid \exists b\, \exists q < \overline{q}\ \big(|b| < n \wedge S = \{x \mid \mathrm{C}^q(x) < n \wedge \mathrm{C}^q(x|b) < k\}\big)\big\}$. Тогда $\mathcal{S}$ состоит из не более чем 2^{n+k} множеств, каждое из которых имеет размер не больше 2^k.*

Доказательство. Ограничение на размер каждого множества $S_{b,q} = \{x \mid \mathrm{C}^q(x|b) < k\}$ получается стандартным рассуждением: существует меньше

2^k условных описаний длины меньше k, значит и описываемых слов меньше, чем 2^k. Ограничение на размер системы $\mathcal{S}$ получается из следующих соображений: разных слов b длины меньше n существует $2^n - 1$ штука, а при фиксированном b разные $S_{b,q}$ вкладываются одно в другое при увеличении q. Значит, разных множеств не больше, чем элементов в самом большом из них. А последнее число меньше 2^k. Перемножая оценки, получаем 2^{n+k}, что и требовалось. □

Теорема 4.8. *Пусть G_n — генератор Нисана–Вигдерсона из следствия 4.6, а $\mathcal{S}$ — система множеств из утверждения 4.7. Тогда существует алгоритм, получающий на вход n, k и $\overline{q}$ и, используя память $O(\overline{q}) + \mathrm{poly}(n)$, находит x, такой что $G_n(x)$ кодирует функцию $F\colon \{0,1\}^n \to \{0,1\}^k$, имеющую не больше $3n$ коллизий для системы $\mathcal{S}$.*

Доказательство. Для вычислений с ограничением на память задача построения эквивалентна задаче распознавания, поэтому мы докажем, что по слову x можно проверить, является ли $G_n(x)$ кодом функции с малым числом коллизий.

В отличие от обычной колмогоровской сложности, сложность с ограничением на память является вычислимой функцией, причём для вычисления сложности C^q, в том числе условной, слова длины n достаточно памяти $O(q+n)$: нужно перебирать все описания по возрастанию длины и запускать на них оптимальный способ описания, ограничивая использованную им память величиной q и контролируя зацикливания. Если получился x, то его сложность вычислена, если получилось другое слово, алгоритм вышел за пределы зоны или зациклился, то нужно перейти к следующему описанию. Для контроля зацикливания нужна удвоенная память, для промежуточных результатов — линейная от n, поэтому в сумме получится $O(q+n)$, как заявлено. Приведём также всевдокод:

Вход: Слова x, b и число q
Выход: Сложность $\mathrm{C}^q(x|b)$
для каждого $z \in \{0,1\}^*$ **выполнить**
  Запустить $U(z,b)$ на зоне q с контролем зацикливания;
  если $U(z,b)$ *корректно вычислено и равно* x **то возвратить** $|z|$;
конец цикла

Алгоритм 4.3: Вычисление условной сложности с ограничением на память.

Далее, заметим, что число коллизий монотонно не убывает при расши-

```
Вход: Числа n, k и q̄
Выход: Слово x, такое что G_n(x) кодирует функцию, имеющую не больше 3n
       коллизий для 𝒮
l := O(n^{2d+6}) ;                          /* Точное значение как в следствии 4.6 */
для каждого x ∈ {0,1}^l выполнить
    для каждого b ∈ {0,1}^{<n} выполнить
        для каждого v ∈ {0,1}^{<n} выполнить
            Запустить U(v,ε) на зоне q̄ с контролем зацикливания;
            если U(v,ε) корректно вычислено то
                y := U(v,ε);
                если C^{q̄}(y|b) < k то
                    count := 1;
                    для каждого w ∈ {0,1}^{<n} выполнить
                        Запустить U(w,ε) на зоне q̄ с контролем зацикливания;
                        если U(w,ε) корректно вычислено то
                            z := U(w,ε);
                            если C^{q̄}(z|b) < k и F(z)=F(y) то count ++;
                            /* Для вычисления F берутся нужные биты G_n(x)  */
                        конец условия
                    конец цикла
                    если count > 3n то продолжить цикл по x;
                конец условия
            конец условия
        конец цикла
    конец цикла
    возвратить x ;   /* Выполняется, только если для всех b и y выполнено
    count ⩽ 3n */
конец цикла
```

Алгоритм 4.4: Поиск хорошего аргумента генератора Нисана–Вигдерсона.

рении множества, поэтому коллизии достаточно посчитать для множеств вида $S_{b,\overline{q}}$ при различных b. Более того, их можно последовательно посчитать для всех b, используя одну и ту же память: дополнительные затраты на хранение b составят $O(n)$. Поэтому достаточно описать процедуру подсчёта числа коллизий для конкретного $S_{b,\overline{q}}$.

Будем последовательно перечислять все элементы $S_{b,\overline{q}}$ и для каждого из них считать число коллизий. Перечислять их можно очень просто: нужно перебирать все описания длины меньше n и для каждого описанного (на зоне $\overline{q}$) слова считать также сложность $\mathrm{C}^{\overline{q}}$ условно на b. Если результат меньше k, соответствующее слово нужно включить в перечисление, иначе пропустить. При этом использованная рабочая память будет равна $O(\overline{q}+n)$.

Наконец, для каждого слова z, полученного в перечислении, нужно по-

считать число коллизий. Вначале нужно вычислить $F(z)$. Это можно сделать, поскольку $F(z)$ является последовательностью конкретных k битов значения генератора $G_n(x)$. По свойству генератора (см. следствие 2.36) эти биты можно вычислить на памяти $\mathrm{poly}(n)$. Далее, нужно завести счётчик коллизий, исходно равный единице, и вновь запустить перечислитель множества $S_{b,\overline{q}}$. Для каждого полученного слова w нужно посчитать значение $F(w)$ и сравнить его с $F(z)$. В случае совпадения нужно увеличить счётчик коллизий на 1. Если к концу перечисления счётчик превысил $3n+1$, то коллизий слишком много, и прообраз x не годится. Если же ни для каких b и z счётчик не был превышен, то прообраз x годится. Алгоритм 4.4 показывает псевдокод.

Описанный алгоритм всегда возвращает верное значение. Поскольку вычисления, требующие памяти $O(\overline{q})$ или $\mathrm{poly}(n)$, не проводятся одновременно, а для хранения промежуточных результатов и осуществления переборов достаточно памяти $O(n)$, общая использованная память равна $O(\overline{q}) + \mathrm{poly}(n)$, как и требовалось. □

4.2.4 Формулировка и доказательство варианта теоремы Мучника

В этом подразделе мы докажем следующий вариант теоремы Мучника:

Теорема 4.9. *Пусть даны двоичные слова a и b, а также числа n, k и s, для которых верно $|b| < n$, $\mathrm{C}^s(a) < n$ и $\mathrm{C}^s(a|b) < k$. Тогда существует такое слово p, что:*

- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(a|p,b) \leqslant O(\log\log s) + O(\log n)$;
- $|p| \leqslant k + O(\log n)$;
- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(p|a) \leqslant O(\log\log s) + O(\log n)$.

Доказательство. Вначале мы докажем теорему в формулировке, где вместо $O(\log\log s)$ стоят $O(\log s)$, а потом покажем, как её усилить.

Идея доказательства состоит в следующем: в качестве p нужно взять значение хеш-функции от a. В качестве хеш-функции нужно взять псевдослучайную функцию, построенную в теореме 4.8. А именно, пусть слово x находится алгоритмом 4.4, получившим на вход n, k и $\overline{q} = s$, т.е. $G_n(x)$ будет кодировать искомую функцию F. Сложность x, таким образом, будет не больше сложности тройки (n,k,s), которая равна $O(\log n + \log s)$.

Поскольку алгоритм нахождения x требует памяти $O(s) + \mathrm{poly}(n)$, сложность x останется такой же и при добавлении такого ограничения на память. Наконец, если x известно, то вычислить значение порождённой им функции на аргументе a можно на памяти $\mathrm{poly}(n)$ в силу теоремы 2.36. Объединив всё вместе, получаем ослабленное третье условие теоремы: $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(p|a) \leqslant O(\log s) + O(\log n)$. Второе условие также выполнено по построению. Осталось доказать первое.

Для доказательства первого условия надо воспользоваться тем, что функция F, закодированная $G_n(x)$, имеет мало коллизий. При известных n, k и s можно найти x, использовав память $O(s) + \mathrm{poly}(n)$. При известных b и s можно перечислять $S_{b,s}$, используя память $O(s+n)$. По условию $a \in S_{b,s}$, а по построению $F(a) = p$ и не более $3n$ элементов $S_{b,s}$ переходят в p под действием F. Поэтому a при известном p можно задать его порядковым номером в перечислении всех прообразов p в множестве $S_{b,s}$. В силу малого числа коллизий этот номер не больше $3n$, поэтому для задания порядкового номера достаточно $O(\log n)$ битов. Сложив полученные оценки, получим, что $C(a|p,b) \leqslant O(\log n) + O(\log s)$. Ограничение на память также будет равно заявленному, поскольку при известном z проверка того, что $F(z) = p$, требует полиномиальной памяти. Таким образом, первое условие также доказано в ослабленном виде.

Чтобы получить теорему в исходной формулировке, достаточно заметить, что в качестве $\overline{q}$ вместо s можно взять $2^{\lceil \log s \rceil}$, т.е. минимальную степень двойки, не меньшую s. Действительно, аргумент x, найденный для такого ограничения $\overline{q}$, подходит и для s, а память, необходимая для поиска x, возрастает не более чем вдвое. Тем самым, оценка $O(s) + \mathrm{poly}(n)$ на эту память остаётся в силе. С другой стороны, сложность такого $\overline{q}$ снижается с $O(\log s)$ до $O(\log \log s)$. Соответствующую замену можно провести в первом и третьем условии, тем самым завершив доказательство теоремы. $\square$

В заключение покажем, что в формулировке теоремы 4.9 можно вообще избавиться от слагаемого $O(\log \log s)$, т.е. верна следующая теорема:

Теорема 4.10. *Пусть даны двоичные слова a и b, а также числа n, k и s, для которых верно $|b| < n$, $\mathrm{C}^s(a) < n$ и $\mathrm{C}^s(a|b) < k$. Тогда существует такое слово p, что:*

- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(a|p,b) \leqslant O(\log n)$;
- $|p| \leqslant k + O(\log n)$;

- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(p|a) \leqslant O(\log n)$.

Доказательство. При $s < 2^{\mathrm{poly}(n)}$ эта формулировка следует из теоремы 4.9. Покажем, что при бóльших s, а именно при $s = 2^{\Omega(n)}$, можно применить теорему 4.1. Действительно, по теореме 2.19 при всех $l < n$ для $d = O(\log n)$ существует $(l,\, 0.25)$-экстрактор $\mathrm{Ext}_l\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^l$. Этот экстрактор можно описать двоичным словом из $l2^{n+d} = 2^{O(n)}$ битов. Если такое слово задано, можно проверить, описывает ли оно экстрактор, используя такую же память. Действительно, для перебора множеств $S \subset \{0,1\}^n$ размера не больше 2^l достаточно зоны 2^{n+l}, для перебора множеств $Y \subset \{0,1\}^l$ достаточно зоны 2^l, а для подсчёта рёбер между S и Y и сравнения этого числа с $|Y|D$ достаточно даже полиномиальной зоны. Таким образом, свойство экстрактора можно распознать, используя память $2^{O(n)} = O(s)$. Значит, нужный экстрактор можно найти, используя такое количество памяти, а значит, его значение можно вычислить. Теперь утверждение теоремы непосредственно следует из теоремы 4.1. □

4.3 Теорема с ограничением на память для нескольких условий

В этом разделе мы докажем вариант теоремы Мучника с несколькими условиями для сложности с ограничением на память. А именно, вначале мы докажем аналоги теорем 4.1 и 4.9 для двух условий, затем выведем аналог теоремы 4.10 и в конце сформулируем теорему для полиномиального числа условий.

4.3.1 Доказательство при помощи явного экстрактора

Мы будем доказывать теорему в следующей формулировке:

Теорема 4.11. *Пусть даны двоичные слова a, b и c, а также числа n, k, l и s, для которых выполнено $\mathrm{C}^s(a) < n$, $\mathrm{C}^s(a|b) < k$ и $\mathrm{C}^s(a|c) < l$. Пусть числа d и r таковы, что существует префиксный $(\max\{k,l\},\, 0.125)$-экстрактор $\mathrm{Ext}_{\max\{k,l\}}\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\max\{k,l\}}$, вычислимый на памяти r. Тогда существуют слова p и q, одно из которых является началом другого, для которых выполнены неравенства:*

- $\mathrm{C}^{O(s+r+n^2)}(a|p,b) \leqslant d + O(\log n)$;

- $\mathrm{C}^{O(s+r+n^2)}(a|q,c) \leqslant d + O(\log n)$;
- $|p| \leqslant k + O(\log n)$;
- $|q| \leqslant l + O(\log n)$;
- $\mathrm{C}^{O(r)}(p|a) \leqslant d + O(\log n)$;
- $\mathrm{C}^{O(r)}(q|a) \leqslant d + O(\log n)$.

Доказательство. Как и в теореме 4.1, будем сначала доказывать ослабленное утверждение, где в первые две условные сложности добавлено s в качестве условия.

Напомним, что слово x называется слабо опасным для множества S, лежащего в левой доле двудольного графа типа (n, m, d), если хотя бы половина соседей x имеют больше $2D$ соседей внутри S. По лемме 3.6 если граф является (k, ε)-экстрактором и $|S| \leqslant K$, то количество слабо опасных слов в S меньше $4\varepsilon K$. Мы будем применять эту лемму к множествам $S_{b,s} = \{x \mid \mathrm{C}^s(x|b) < k\}$ и $T_{c,s} = \{x \mid \mathrm{C}^s(x|c) < l\}$. Как и в теореме 4.1, нельзя доказать, что a не является слабо опасным в множествах $S_{b,s}$ и $T_{c,s}$, поэтому мы используем отдельную итеративную конструкцию для слабо опасных слов. Трудность состоит в том, что слово a должно не быть слабо опасным одновременно для двух разных множеств, причём опасность измеряется относительно двух разных экстракторов, один из которых является префиксом другого. Конструкция будет похожа на конструкцию из теоремы 4.1, со следующим отличием: в качестве экстрактора с меньшим k мы всегда будем брать префикс исходного. Опишем процесс нахождения этих двух множеств формально.

Обозначим через k_i и l_j числа $k - i$ и $l - j$ соответственно. Для $\kappa < \max\{k, l\}$ обозначим через Ext_κ префикс экстрактора $\mathrm{Ext}_{\max\{k,l\}}$ длины κ. Заметим, что этот префикс вычисляется на памяти r при любом заданном κ. (Действительно, на такой памяти можно вычислить значение всего экстрактора, а взятие префикса требует совсем небольшой памяти). Далее, введём обозначения $S^{(0)}_{b,s} = S_{b,s}$ и $T^{(0)}_{c,s} = T_{c,s}$. По индукции определим $S^{(i+1)}_{b,s}$ как множество слабо опасных слов для множества $S^{(i)}_{b,s}$ в экстракторе Ext_{k_i} и $T^{(j+1)}_{c,s}$ как множество слабо опасных слов для множества $T^{(j)}_{c,s}$ в экстракторе Ext_{l_j}. В силу леммы 3.6 и выбора ε можно по индукции доказать, что $\left|S^{(i)}_{b,s}\right| < K/2^i = 2^{k_i}$ и $\left|T^{(j)}_{c,s}\right| < L/2^j = 2^{l_j}$. Поскольку при $i = k$ и $j = l$ величины k_i и l_j равны нулю, получаем, что множества $S^{(k)}_{b,s}$ и $T^{(l)}_{c,s}$

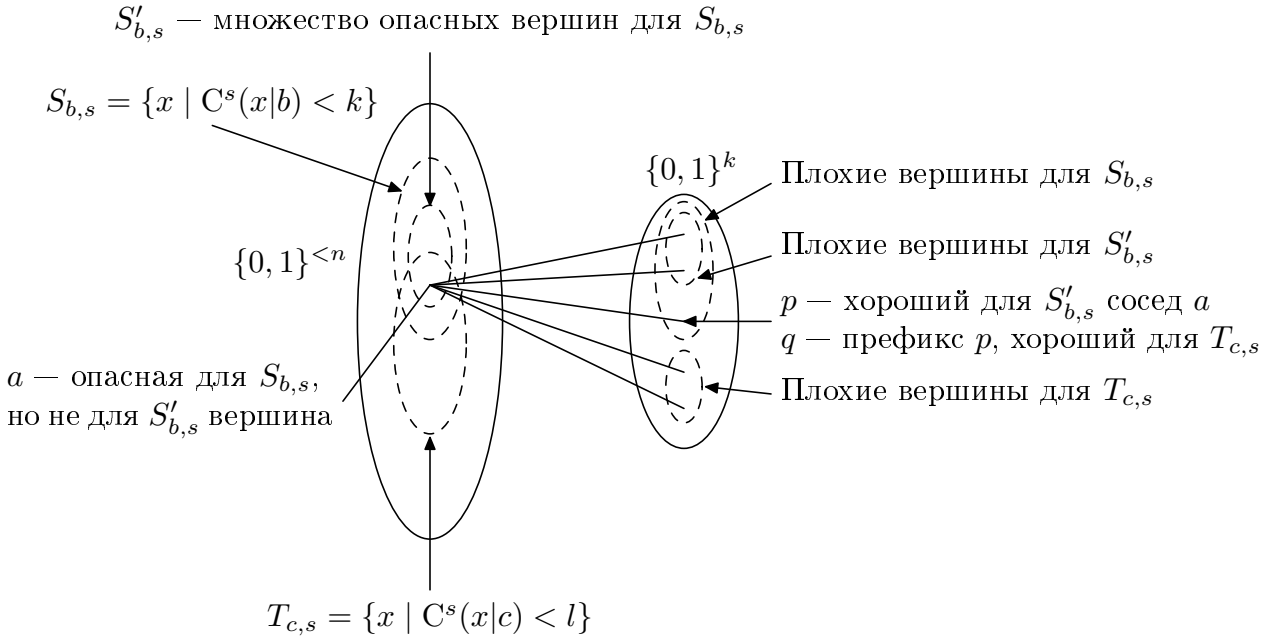


Рис. 4.3: Доказательство теоремы Мучника с ограничением на память: выбор p и q для опасного a.

пусты. Значит, найдутся такие i и j, что $a \in S^{(i)}_{b,s} \setminus S^{(i+1)}_{b,s}$ и $a \in T^{(j)}_{c,s} \setminus T^{(j+1)}_{c,s}$. Зафиксируем эти i и j для дальнейшего.

Поскольку слово a не является слабо опасным для множества $S^{(i)}_{b,s}$ в экстракторе Ext_{k_i}, то больше половины соседей a в этом экстракторе имеют меньше $2D$ соседей внутри множества $S^{(i)}_{b,s}$. А поскольку a не является слабо опасным и для множества $T^{(j)}_{c,s}$ в экстракторе Ext_{l_j}, то больше половины соседей a в этом экстракторе имеют меньше $2D$ соседей внутри множества $T^{(j)}_{c,s}$. Поскольку экстракторы Ext_{k_i} и Ext_{l_j} являются префиксами экстрактора $\mathrm{Ext}_{\max\{k_i,l_j\}}$, можно заключить, что существует сосед a в последнем экстракторе со следующими свойствами: его префикс длины k_i имеет меньше $2D$ соседей внутри множества $S^{(i)}_{b,s}$ в экстракторе Ext_{k_i}, а его префикс длины l_j имеет меньше $2D$ соседей внутри множества $T^{(j)}_{c,s}$ в экстракторе Ext_{l_j}. Обозначим префикс этого соседа длины k_i через p, а префикс длины l_j — через q. Ясно, что одно из слов p и q будет префиксом другого, а более длинное из них будет префиксом исходного соседа. Покажем, что эти p и q искомые.

Действительно, условие на длины p и q получаются по построению: $|p| = k - i \leqslant k$ и $|q| = l - j \leqslant l$. Оценка на условные сложности p и q при известном a также проста: нужно задать n и k_i (соответственно, n и l_j), на

что нужно $O(\log n)$ битов, и номер p (соответственно, q) среди соседей a в экстракторе Ext_{k_i} (соответственно, Ext_{l_j}), на что нужно d битов. Поскольку эти экстракторы вычисляются на памяти r, получаем нужное ограничение на память. Суммируя, получаем $\mathrm{C}^{O(r)}(p|a) \leqslant d + O(\log n)$ и $\mathrm{C}^{O(r)}(q|a) \leqslant d + O(\log n)$, что и требовалось.

Теперь построим оценки на сложности a при условиях p, b и s и при условиях q, c и s. Оценки получаются одинаковым способом, поэтому опишем подробно только первую. Заметим, что при известных ограничении на зону s, слове b и индексе i можно перечислять $S^{(i)}_{b,s}$, используя память $O(s+r+n^2)$. Этот факт доказывается точно так же, как аналогичное утверждение в теореме 4.1. Далее, для описания a необходимо указать n, k, i и номер a среди соседей p в множестве $S^{(i)}_{b,s}$ для экстрактора Ext_{k_i}. В силу того, что p не является слабо опасным для a, описание указанного номера требует не более $d+1$ бита, а для задания остальных параметров достаточно $O(\log n)$ битов. На памяти $O(s+r+n^2)$ можно перечислять множество $S^{(i)}_{b,s}$ и для каждого полученного слова w вычислять все значения экстрактора (на памяти r, причём можно использовать тот же участок памяти), затем проверять, встречается ли p среди них, и включать w в перечисление, если встречается. Слово a будет задано своим номером в этом перечислении, поэтому $\mathrm{C}^{O(s+r+n^2)}(a|p,b,s) \leqslant d + O(\log n)$, как и было заявлено.

Избавиться от параметра s в последней сложности, можно так же, как в теореме 4.1: нужно перечислять множество $S^{(i)}_b = \cup_{s=1}^{\infty} S^{(i)}_{b,s}$ при помощи алгоритма 4.2. Тогда каждое слово в перечислении встретится ровно один раз и к моменту перечисления всех слов из $S^{(i)}_{b,s}$ будет использована память $O(s+r+n^2)$. При использовании такого перечисления больше нет необходимости добавлять s в описание a, таким образом теорема доказывается в исходной формулировке. □

Завершим подраздел следствием из теоремы 4.3, полученным с использованием известных конструкций экстракторов:

Следствие 4.12. *Пусть даны двоичные слова a, b и c, а также числа n, k, l и s, для которых выполнено $\mathrm{C}^s(a) < n$, $\mathrm{C}^s(a|b) < k$ и $\mathrm{C}^s(a|c) < l$. Тогда существуют слова p и q, одно из которых является началом другого, для которых выполнены неравенства:*

- $\mathrm{C}^{O(s+\mathrm{poly}(n))}(a|p,b) \leqslant O(\log^3 n)$;
- $\mathrm{C}^{O(s+\mathrm{poly}(n))}(a|q,c) \leqslant O(\log^3 n)$;

- $|p| \leqslant k + O(\log n)$;
- $|q| \leqslant l + O(\log n)$;
- $\mathrm{C}^{\mathrm{poly}(n)}(p|a) \leqslant O(\log^3 n)$;
- $\mathrm{C}^{\mathrm{poly}(n)}(q|a) \leqslant O(\log^3 n)$.

Доказательство. Утверждение получается непосредственной подстановкой экстрактора, существующего по следствию 2.22, в теорему 4.3. □

Замечание 4.13. Аналогично теоремам с одним условием, можно добиться, чтобы p было длины ровно k, а q было длины ровно l, либо чтобы $\mathrm{C}^{O(s+\mathrm{poly}(n))}(a|p,b) \leqslant O(1)$ и $\mathrm{C}^{O(s+\mathrm{poly}(n))}(a|q,c) \leqslant O(1)$, но при этом $|p| \leqslant k+O(\log^3 n)$ и $|q| \leqslant l+O(\log^3 n)$. В обоих случаях растут $\mathrm{C}^{\mathrm{poly}(n)}(p|a)$ и $\mathrm{C}^{\mathrm{poly}(n)}(q|a)$, оставаясь порядка $\log^3 n$.

4.3.2 Доказательство при помощи генератора Нисана–Вигдерсона

Здесь мы докажем теорему Мучника для сложности с ограничением на память и двух условий при помощи генератора Нисана–Вигдерсона. Общий ход рассуждений будет повторять изложение раздела 4.2, поэтому мы опустим некоторые подробности. Нам вновь будет достаточно одной хеш-функции: мы сформулируем нужное свойство этой функции, затем докажем, что оно распознаётся схемами константной глубины и полиномиального размера, из чего сделаем вывод, что объект с нужным свойством встречается среди значений генератора.

Пусть заданы некоторая функция $F\colon \{0,1\}^n \to \{0,1\}^k$ и число $l \leqslant k$. Пусть также заданы система множеств $\mathcal{S}$, состоящая из не более чем 2^{n+k} подмножеств $\{0,1\}^n$ размера не более 2^k, и система множеств $\mathcal{T}$, состоящая из не более чем 2^{n+l} подмножеств $\{0,1\}^n$ размера не более 2^l. Будем говорить, что F имеет меньше m коллизий относительно $\mathcal{S}$ и $\mathcal{T}$, если для любого $x \in \{0,1\}^k$ и любого $S \in \mathcal{S}$ элемент x имеет менее m прообразов внутри S, а также для любого $y \in \{0,1\}^l$ и любого $T \in \mathcal{T}$ множество $y \cdot \{0,1\}^{k-l}$ имеет менее m прообразов внутри T.

Докажем следующую теорему существования, аналогичную теореме 4.4:

Теорема 4.14. *Существует полином $p(n)$, такой что для любых $l \leqslant k \leqslant n$ и любых систем $\mathcal{S}$ и $\mathcal{T}$, состоящих из не более чем 2^{n+k} (соотв., 2^{n+l}) подмножеств $\{0,1\}^n$, каждое из которых имеет размер не больше 2^k*

(соотв., 2^l), хотя бы половина всех функций $F\colon \{0,1\}^n \to \{0,1\}^k$ имеют меньше $p(n)$ коллизий относительно $\mathcal{S}$ и $\mathcal{T}$.

Доказательство. Для доказательства вновь используем вероятностный метод. А именно, докажем, что случайная функция, значения которой выбраны равномерно и независимо, обладает нужным свойством с вероятностью не меньше $1/2$. Для этого оценим сверху вероятность обратного события.

Если функция F не соответствует условию для $\mathcal{S}$ и $\mathcal{T}$, то для некоторого $S \in \mathcal{S}$ или для некоторого $T \in \mathcal{T}$ она имеет хотя бы m коллизий. Вероятность того, что коллизий хотя бы m для фиксированного $S \in \mathcal{S}$, не превышает $2^k C_{2^k}^m \left(\frac{1}{2^k}\right)^m$: здесь 2^k — количество всевозможных x, $C_{2^k}^m$ — верхняя оценка количества всевозможных m-элементых подмножеств S, $\left(\frac{1}{2^k}\right)^m$ — вероятность того, что на всех m элементах подмножества функция равна x. Используя неравенство $C_{2^k}^m \leqslant \frac{\left(2^k\right)^m}{m!}$, полученную вероятность можно оценить величиной $\frac{2^k}{m!}$. Аналогично, вероятность того, что коллизий хотя бы m для фиксированного $T \in \mathcal{T}$, не превышает $2^l C_{2^l}^m \left(\frac{2^{k-l}}{2^k}\right)^m = 2^l C_{2^l}^m \left(\frac{1}{2^l}\right)^m$. Здесь $\frac{2^{k-l}}{2^k}$ — вероятность того, что значение функции попадёт в множество $y \cdot \{0,1\}^{k-l}$ для фиксированного y. Применив аналогичное неравенство для числа сочетаний, оценим эту вероятность величиной $\frac{2^l}{m!}$. Просуммировав теперь по всем S и T, получаем оценку $2^{n+k}\frac{2^k}{m!} + 2^{n+l}\frac{2^l}{m!} \leqslant \frac{2^{n+2k+1}}{m!}$. Если взять $m = 4n$, то эта величина меньше одной второй (при $n > 1$), значит вероятность того, что F не подходит, меньше одной второй, а значит, больше половины функций F подходят, что и требовалось. □

Далее, докажем, что свойство малого числа коллизий распознаётся схемой константной глубины и полиномиального размера. Эта схема практически полностью повторяет схему, описанную в доказательстве теоремы 4.5, поэтому мы не будем приводить её полностью, а опишем кратко. Малое число коллизий в каждом $S \in \mathcal{S}$ проверяется точно так же, как и в теореме 4.5 (см. рис. 4.2). Малое число коллизий в каждом $T \in \mathcal{T}$ проверяется с небольшим отличием: в самом начале вместо совпадения значений $F(x_i)$ и $F(x_j)$ проверяется, что совпадают их префиксы длины l. Для этого нужно брать конъюнкцию эквиваленций не по всем битам, а только по первым l. На следующих этапах схема выглядит точно так же, как и раньше. Полиномиальный размер, константная глубина и корректность работы схемы

также доказываются аналогично.

Как следствие, по лемме 2.37 получим, что хеш-функции с малым числом коллизий относительно $\mathcal{S}$ и $\mathcal{T}$ не только существуют, но и встречаются среди значений генератора Нисана–Вигдерсона.

Теперь в качестве систем $\mathcal{S}$ и $\mathcal{T}$ мы возьмём системы

$$\left\{S \mid \exists b \exists q < \overline{q}\ \left(|b| < n \wedge S = \{x \in \{0,1\}^n \mid \mathrm{C}^q(x|b) < k\}\right)\right\}$$

и

$$\left\{T \mid \exists c \exists q < \overline{q}\ \left(|c| < n \wedge T = \{x \in \{0,1\}^n \mid \mathrm{C}^q(x|c) < l\}\right)\right\}$$

соответственно. Аналогично утверждению 4.7 легко показать, что эти системы удовлетворяют нужным ограничениям на размеры.

Аналогично теореме 4.8 можно доказать, что аргумент x, при котором значение генератора Нисана–Вигдерсона $G_n(x)$ кодирует функцию, у которой меньше $4n$ коллизий относительно $\mathcal{S}$ и $\mathcal{T}$, можно найти на памяти $O(\overline{q}) + \mathrm{poly}(n)$. Алгоритм будет почти полностью повторять алгоритм 4.4 с одним отличием: считая коллизии внутри множества $T \in \mathcal{T}$, в строке 14 нужно сравнивать не значения F целиком, а их префиксы длины l.

Наконец, докажем теорему Мучника для двух условий в следующей формулировке:

Теорема 4.15. *Пусть даны двоичные слова a, b и c, а также числа n, k, l и s, для которых выполнено $|b| < n$, $|c| < n$, $\mathrm{C}^s(a) < n$, $\mathrm{C}^s(a|b) < k$ и $\mathrm{C}^s(a|c) < l$. Тогда существуют слова p и q, одно из которых является началом другого, для которых выполнены неравенства:*

- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(a|p,b) \leqslant O(\log\log s) + O(\log n)$;
- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(a|q,c) \leqslant O(\log\log s) + O(\log n)$;
- $|p| \leqslant k + O(\log n)$;
- $|q| \leqslant l + O(\log n)$;
- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(p|a) \leqslant O(\log\log s) + O(\log n)$;
- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(q|a) \leqslant O(\log\log s) + O(\log n)$.

Доказательство. Общий ход рассуждений повторяет доказательство теоремы 4.9. Без ограничения общности будем считать, что $l \leqslant k$. Используя $O(\log n) + O(\log s)$ битов, можно описать аргумент x, на котором генератор возвращает функцию с малым числом коллизий. Слово p будет значением этой функции на слове a, а слово q — префиксом слова p длины l. При этом

памяти $O(s) + \mathrm{poly}(n)$ достаточно, чтобы найти и нужный аргумент генератора, и значение сгенерированной функции на слове a. Таким образом, доказаны условия 3–6 (с заменой $O(\log\log s)$ на $O(\log s)$). Первое условие следует из того, что слово p имеет мало прообразов внутри $S_{b,s}$, поэтому при известных b, s и заданной хеш-функции можно задать a его номером в перечислении этих прообразов. Второе условие следует из того, что множество всех продолжений q имеет мало прообразов внутри $T_{c,s}$, поэтому при известных c, s и хеш-функции слово a вновь можно задать его номером среди этих прообразов. Таким образом, условия 1–2 также доказаны с заменой $O(\log\log s)$ на $O(\log s)$.

Наконец, можно заменить $O(\log s)$ на $O(\log\log s)$, если в качестве ограничения на память вместо s брать минимальную степень двойки, не меньшую s. □

4.3.3 Компиляция двух подходов и теорема для полиномиального числа условий

Как и раньше, можно объединить два подхода и доказать такой аналог теоремы 4.10.

Теорема 4.16. *Пусть даны двоичные слова a, b и c, а также числа n, k, l и s, для которых выполнено $|b| < n$, $|c| < n$, $\mathrm{C}^s(a) < n$, $\mathrm{C}^s(a|b) < k$ и $\mathrm{C}^s(a|c) < l$. Тогда существуют слова p и q, одно из которых является началом другого, для которых выполнены неравенства:*

- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(a|p,b) \leqslant O(\log n)$;
- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(a|q,c) \leqslant O(\log n)$;
- $|p| \leqslant k + O(\log n)$;
- $|q| \leqslant l + O(\log n)$;
- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(p|a) \leqslant O(\log n)$;
- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(q|a) \leqslant O(\log n)$.

Доказательство. Для ограничений на зону, меньших $2^{\mathrm{poly}(n)}$, можно применить теорему 4.15: в этом случае $O(\log\log s) = O(\log n)$. Для бóльших ограничений на зону можно применить теорему 4.11: используя память $2^{\mathrm{poly}(n)}$, можно вычислить оптимальный префиксный экстрактор. □

Наконец, можно распространить теорему на полиномиальное число условий. В этом случае работают оба подхода: и использование явного префиксного экстрактора, и использование генератора псевдослучайных чисел. В первом случае итеративная конструкция проводится отдельно для каждого условия, во втором случае у хеш-функции должно быть мало коллизий для каждого из множеств простых слов. Более того, при использовании генератора можно доказать аналог теоремы не только для полиномиального, но и для экспоненциального числа условий. Поскольку последний результат достаточно неожиданный, то мы его опишем подробно, а первый подход разберём кратко.

Итак, при помощи явных экстракторов можно доказать следующий аналог теоремы 3.7:

Теорема 4.17. *Пусть даны числа n, s, $r = \mathrm{poly}(n)$ и $k_1, \ldots, k_r$ и двоичные слова a, b_1, …, b_r, такие что $\mathrm{C}^s(a) < n$ и $\mathrm{C}^s(a|b_i) < k_i$ при всех $i = 1, \ldots, r$. Пусть числа d и q таковы, что существует префиксный $(\max_i\{k_i\}, \frac{1}{4r})$-экстрактор $\mathrm{Ext}_{\max_i\{k_i\}} \colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\max_i\{k_i\}}$, вычислимый на памяти q. Тогда существуют слова p_1, …, p_r, для которых выполнены следующие условия:*

- *Все p_i являются префиксами одного и того же слова;*
- $\mathrm{C}^{O(s+q+n^2)}(a|p_i, b_i) \leqslant d + O(\log n)$ *при всех* $i = 1, \ldots, r$*;*
- $|p_i| \leqslant k_i + O(\log n)$ *при всех* $i = 1, \ldots, r$*;*
- $\mathrm{C}^{O(q)}(p_i|a) \leqslant d + O(\log n)$ *при всех* $i = 1, \ldots, r$.

Идея доказательства остаётся прежней. Для начала нужно переопределить понятие слабо опасного слова: слово будет таковым, если хотя бы доля $\frac{1}{r}$ его соседей плохие. Далее, для каждого b_i строится иерархия множеств слабо опасных слов $S^{(j)}_{b_i,s}$ и доказывается, что для каждого i на какой-то итерации слово a перестаёт быть слабо опасным. Значит, у a найдётся сосед, который будет хорошим для всех b_i на соответствующей итерации. Его префиксы нужных длин и будут описаниями p_i. Тогда p_i при известном a описывается номером итерации и номером p_i среди соседей a в экстракторе, а слово a при известных p_i и b_i описывается своим номером в перечислении соседей p_i в множестве $S^{(j)}_{b_i,s}$.

Отметим, что при использовании известных явных экстракторов для полиномиальной памяти условные сложности во втором и четвёртом условиях получатся равными $O(\log^3 n)$.

При помощи же генераторов псевдослучайных чисел можно доказать такую теорему:

Теорема 4.18. *Пусть даны числа n, s и $r = \mathrm{poly}(n)$ и двоичное слова a, такие что $\mathrm{C}^s(a) < n$. Тогда существует слово p длины n, для которого выполнены следующие условия:*

- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(a|p_b, b) \leqslant O(\log\log s) + O(\log n)$ *при любом b длины не больше r, где p_b — префикс p длины $\mathrm{C}^s(a|b)$;*
- $\mathrm{C}^{O(s)+\mathrm{poly}(n)}(p|a) \leqslant O(\log\log s) + O(\log n)$.

Таким образом, применение генераторов не только уменьшает с $O(\log^3 n)$ до $O(\log n)$ оценки на условные сложности при полиномиальном ограничении на память, но и увеличивает максимальное число условий с полиномиального до экспоненциального (все слова длины $\mathrm{poly}(n)$). Такого эффекта нет при сложности без ограничений, поскольку сложность без ограничений не вычислима. Проведя доказательство, мы ещё раз обсудим, почему оно не проходит для сложности без ресурсных ограничений.

Доказательство. Доказательство состоит из стандартных шагов. Во-первых, мы определим нужное свойство хеш-функции и докажем вероятностным методом, что такие функции существуют. Во-вторых, мы докажем, что это свойство можно распознать схемой константной глубины. Из этого мы сделаем вывод, что функции, обладающие этим свойством, можно породить генератором Нисана–Вигдерсона. В-третьих, мы докажем, что нужный аргумент генератора можно найти, используя небольшую память. Наконец, в-четвёртых, мы выведем доказательство теоремы из этой конструкции.

Нужное свойство хеш-функции сформулируем так: для всех слов b длины не больше r и всех $k \leqslant n$ размер всех коллизий в множестве $S_{b,k,s} = \{x \mid \mathrm{C}^s(x|b) < k\}$ не больше m. При этом под коллизией будем понимать набор $\{x_1, \ldots, x_\mu\}$, такой что префиксы длины k совпадают у всех $F(x_i)$.

Существование функций с заданным свойством доказывается стандартным применением вероятностного метода. Вероятность коллизии размера m в одном множестве $S_{b,k,s}$ не превосходит $C_{2^k}^m \left(\frac{1}{2^k}\right)^m$. Эта вероятность оценивается сверху величиной $\frac{1}{m!} < 2^{-m}$. Если взять m достаточно большим, например больше $r + \log n$, то и сумма таких вероятностей по всем b и k

будет меньше 1. Значит, с положительной вероятностью случайная хеш-функция не содержит коллизий размера m ни в одном из множеств $S_{b,k,s}$.

Существование схемы константной глубины, проверяющей малый размер коллизий в конкретном множестве, мы уже доказывали. Важно, что длина b ограничена $\mathrm{poly}(n)$, поэтому число всевозможных пар (b,k) равно $2^{\mathrm{poly}(n)}$. Это больше, чем полином от длины записи функции, которая есть $n2^n$, но можно искусственно дополнить код функции ненужными символами, чтобы его длина стала равна $2^{\mathrm{poly}(n)}$. Тогда размер схемы — конъюнкции ранее построенных схем — станет полиномиальным от длины входа, а глубина останется константной. Длина аргумента генератора Нисана–Вигдерсона будет равна $O\left((\log(2^{\mathrm{poly}(n)}))^{2d+6}\right) = O\left(\mathrm{poly}(n)^{2d+6}\right) = \mathrm{poly}(n)$.[4]

Применяя лемму 2.37, получаем, что среди значений генератора Нисана–Вигдерсона на аргументах полиномиальной длины есть код функции с малым числом коллизий во всех множествах $S_{b,k,s}$.

Далее, нужный аргумент генератора можно найти, используя память $O(s)+\mathrm{poly}(n)$. Эквивалентно, по аргументу генератора можно распознать, является ли значение на нём кодом функции с малым числом коллизий. Действительно, будем перебирать все слова b длины не больше r и все числа $k \leqslant n$. Для каждой пары (b,k) будем перечислять $S_{b,k,s}$ и для каждого вновь полученного слова вычислять размер коллизии с его участием. Для этого нужно вычислить значение сгенерированной функции на этом слове и сохранить префикс этого значения длины k. Затем нужно вновь запустить перечисление $S_{b,k,s}$, на получаемых словах вычислять значение функции и проверять, совпадает ли его префикс длины k с сохранённым. В случае совпадения нужно увеличить счётчик коллизий на единицу. Если хотя бы для одной пары (b,k) хотя бы для одного элемента $S_{b,k,s}$ счётчик превысит m, то аргумент генератора следует отвергнуть. Если же для всех пар (b,k) и всех элементов $S_{b,k,s}$ счётчик остался меньше m, то аргумент нужно принять. При этом вычисления, требующие зоны $O(s)$, можно вести на одном и том же участке памяти, а все накладные расходы требуют зоны $\mathrm{poly}(n)$. Таким образом, лексикографически первый аргумент генератора, порождающий нужную функцию, имеет сложность $O(\log n + \log\log s)$ при ограничении на память $O(s)+\mathrm{poly}(n)$. Добавка $O(\log\log s)$ происходит от

[4] Аналогично можно усилить формулировки теорем 4.9, 4.10, 4.15 и 4.16, заменив условия $|b| < n$ и $|c| < n$ на $|b| < \mathrm{poly}(n)$ и $|c| < \mathrm{poly}(n)$, соответственно.

того, что для нахождения этого аргумента нужно знать s с точностью до умножения на 2.

Теперь покажем, как доказать утверждение теоремы. В качестве p нужно взять $F(a)$, где F — функция, полученная при помощи генератора из нужного аргумента. Второе условие соблюдено, поскольку аргумент генератора, порождающий функцию F, имеет логарифмическую сложность. Если известны этот аргумент и a, то p находится на полиномиальной памяти. Первое же условие выполнено в силу условия на малость числа коллизий у F. Помимо b и p_b, для нахождения a нужно задать аргумент генератора, порождающий F, и номер a среди элементов $S_{b,k_b,s}$, образы которых начинаются с p_b. (Мы обозначаем через k_b величину $\mathrm{C}(a|b)$). В силу малого числа коллизий внутри $S_{b,k_b,s}$ для записи этого номера достаточно логарифмического числа битов. Памяти $O(s) + \mathrm{poly}(n)$ будет достаточно и для поиска нужного аргумента генератора, и для нахождения a: на такой памяти можно перечислять $S_{b,k_b,s}$ и считать значения F на всех перечисленных элементах. Соответственно, можно перечислять и те элементы $S_{b,k_b,s}$, значение F на которых начинается с p_b. Слово a определится номером в этом перечислении. □

Обсудим, почему аналогичное рассуждение не подойдёт для исходной теоремы. Первые два шага доказательства останутся в силе: найдётся хеш-функция с малым числом коллизий во всех множествах $S_{b,k} = \{x \mid \mathrm{C}(x|b) < k\}$, и её можно распознать схемой константной глубины. Однако, третий шаг провести не получится: непонятно, почему одна из таких схем имеет малую сложность. Из-за того, что функция C не вычислима, по множеству нельзя алгоритмически понять, имеет оно вид $S_{b,k}$ или не имеет. Поэтому нельзя проверить малость размера коллизий только на множествах $S_{b,k}$, разрешив коллизиям на других множествах быть большими. Из-за этого приходится вводить несколько хеш-функций, требовать малого числа коллизий во всех множествах, и возникает исходная конструкцию Мучника.

Отметим, что теорему 4.18 можно неформально интерпретировать как способ универсального кодирования: по слову a получается слово p, такое что для любого не слишком длинного b достаточно прочесть первые $\mathrm{C}^s(a|b)$ битов p, чтобы затем восстановить a. В терминах коммуникации с подсказками можно сказать, что Алиса не нуждается в подсказке, зависящей от b, а должна лишь знать максимально возможную длину b и ширину канала

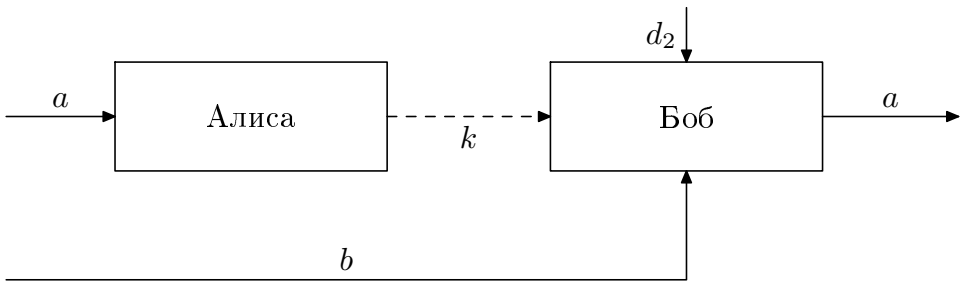


Рис. 4.4: Схема коммуникации с подсказкой для Боба: Алиса производит универсальный код p, Боб получает первые k битов этого кода, слово b и подсказку d_2 и восстанавливает a.

(см. рис. 4.4).

Также можно сформулировать теорему в терминах извлечения случайности: по слову a получается не просто несжимаемое на полиномиальной памяти описание p, а такое, что для любого слова b префикс p длины $\mathrm{C}^s(a|b)$ также не сжимаем при известном b.

Завершим раздел метафорическим описанием теоремы, принадлежащим А.Шеню. Докладчик делает доклад для аудитории с разным уровнем подготовки. Разные слушатели уже знают некоторую часть того, что докладчик хочет рассказать. Тогда доклад может быть выстроен таким образом, чтобы каждый слушатель мог уйти, прослушав минимально необходимую часть доклада и после небольшой подсказки восстановить всю исходную информацию.

Глава 5

Теорема Мучника для CAM-сложности с ограничением на время

Время как вычислительный ресурс отличается от памяти одним существенным недостатком: его нельзя использовать вторично. Поэтому никакие переборные конструкции не могут сработать при доказательстве теоремы с ограничением на время, можно использовать только явные конструкции. Тем не менее, даже с использованием явных конструкций не удаётся получить вариант теоремы Мучника для обычной сложности. Зато можно получить вариант теоремы Мучника для CAM-сложности. Здесь используется явная конструкция экстракторов Тревисана [49], а само рассуждение во многом использует идеи, изложенные в статье [13]. Поскольку мы будем говорить только об ограничениях на время, то верхний индекс в обозначениях для сложности в этой и только этой главе будет означать ограничение на время, а не на память.

5.1 Формулировка теоремы

Мы докажем следующее утверждение:

Теорема 5.1. *Для любого полинома $t_1(n)$ существует полином $t_2(n)$, для которого выполнено следующее свойство. Пусть даны числа n и k, слово a длины меньше n и слово b, такие что $\mathrm{C}^{t_1(n)}(a|b) < k$. Тогда существует слово p, удовлетворяющее следующим условиям:*

- $\mathrm{CAM}^{t_2(n)}(a|b,p) = O(\log^3 n)$;
- $|p| \leqslant k + O(\log^3 n)$;
- $\mathrm{C}^{t_2(n)}(p|a) = O(\log^3 n)$.

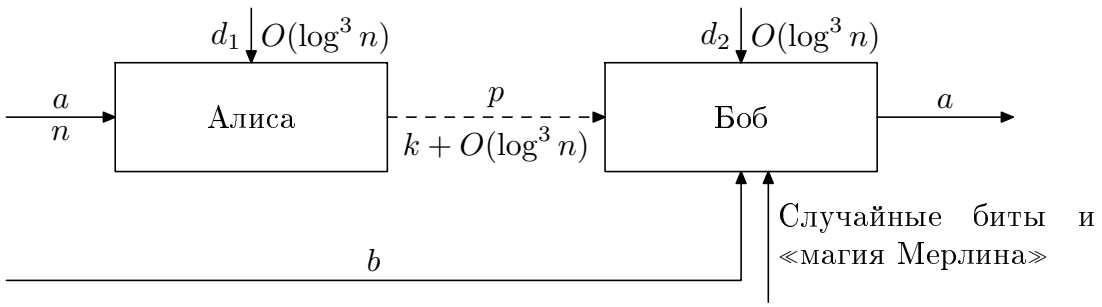


Рис. 5.1: Схема коммуникации в теореме для CAM-сложности.

В терминах задачи о передаче информации эту теорему можно интерпретировать так: оптимальное условное кодирование возможно, если кодировщик и декодировщик полиномиально ограничены, но оба могут получать подсказки длины $O(\log^3 n)$, а декодировщик может к тому же использовать случайные биты и «магию Мерлина» (см. рис. 5.1).

Можно заметить, что в некоторых местах $O(\log^3 n)$ можно заменить на $O(1)$. Так, биты, необходимые для декодирования a по b и p, можно включить в программу p и передавать кодировщику при построении p. Из-за этого увеличатся константы в $O(\cdot)$-обозначениях во втором и третьем соотношениях, а в первом $O(\log^3 n)$ заменится на $O(1)$. Можно сделать наоборот: лишние биты в слове p перенести в описание a при известных b и p. Тогда слово p будет иметь длину ровно k, а константы в остальных $O(\cdot)$-обозначениях увеличатся. Только сложность p при известном a нельзя уменьшить в данной конструкции.

5.2 Описание конструкции

Вначале дадим несколько определений, которые используются при построении функции Тревисана.

Определение 5.2. Пусть l и d суть натуральные числа, а $S_i \subset \{1, \ldots, d\}$ для $i = 1, \ldots, m$. Система множеств $\{S_1, \ldots, S_m\}$ называется *(l, d)-слабым дизайном*, если каждое S_i состоит из l элементов и для каждого $i > 1$ сумма $\sum_{j=1}^{i-1} 2^{|S_i \cap S_j|}$ не превосходит $m - 1$.

Теорема 5.3 ([37]). *Существует алгоритм, работающий полиномиальное от $l + m$ время и генерирующий (l, d)-слабый дизайн для $d = O(l^2 \log m)$.*

Для дальнейшего мы зафиксируем некоторый слабый дизайн, порождённый этим алгоритмом. Для слова $x \in \{0,1\}^d$ через $x\big|_{S_i}$ будем обозначать проекцию слова x на координаты, входящие в S_i. Иными словами, если $x = x_1 \dots x_d$, а $S_i = \{j_1, \dots, j_l\}$, где $j_1 < \dots < j_l$, то $x\big|_{S_i} = x_{j_1} \dots x_{j_l}$.

Второй ключевой компонент конструкции — декодируемые списком коды, исправляющие ошибки. Мы будем говорить о хэмминговском расстоянии между словами:

Определение 5.4. Пусть слова v и v' лежат в множестве $\{0,1\}^n$. Тогда *хэмминговским расстоянием* между ними называется доля несовпадающих битов, т.е. $\big|\{i \mid v_i \neq v'_i\}\big|/n$. Если хэмминговское расстояние от v до v' не больше $(1-\alpha)$, то мы также будем говорить, что каждое из слов является *α-аппроксимацией* другого.

Нам потребуются коды, существование которых утверждает следующая теорема:

Теорема 5.5 ([47]). *Для любого натурального $n > 0$ и действительного $\delta > 0$ существуют $\bar{n} = \mathrm{poly}(n/\delta)$ и функция $\mathrm{LDC}_{n,\delta}\colon \{0,1\}^n \to \{0,1\}^{\bar{n}}$, вычислимая за полиномиальная от n/δ время, для которой существует алгоритм декодирования списком. Этот алгоритм работает полиномиальное от n/δ время и по любому $y \in \{0,1\}^{\bar{n}}$ возвращает список всех $x \in \{0,1\}^n$, таких что $\mathrm{LDC}_{n,\delta}(x)$ является $\left(\frac{1}{2}+\delta\right)$-аппроксимацией y.*

Из теоремы, в частности, можно сделать вывод, что для любого y найдётся не более полинома от n/δ слов x, таких что $\mathrm{LDC}_{n,\delta}(x)$ является $(\frac{1}{2}+\delta)$-аппроксимацией y. Заметим, что без ограничения общности можно считать, что $\bar{n}$ есть степень двойки (в случае, если это не так, дополним все слова нулями). В дальнейшем будем использовать обозначение $l = l(n) = \log \bar{n}$. Таким образом, для каждого $u \in \{0,1\}^n$ слово $\mathrm{LDC}_{n,\delta}(u)$ длины 2^l можно интерпретировать как таблицу истинности некоторой булевой функции $\hat{u}\colon \{0,1\}^l \to \{0,1\}$.

Определение 5.6. Пусть фиксированы слабый дизайн $(S_1, \dots, S_m)$ и код $\mathrm{LDC}_{n,\delta}$, декодируемый списком. Определим *функцию Тревисана* $\mathrm{TR}_\delta\colon \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ следующим равенством:

$$\mathrm{TR}_\delta(u, y) = \hat{u}(y\big|_{S_1}) \dots \hat{u}(y\big|_{S_m}).$$

В работе [49] показано, что функция Тревисана является экстрактором, но мы этим пользоваться не будем, вместо этого воспользуемся определением напрямую. Начнём с определения значений параметров. Числа n и k заданы формулировкой теоремы. Затем, числа m, d, l и δ выбираются так, чтобы были одновременно выполнены следующие условия:

- $m = k + d + 1$;
- $\delta = \frac{1}{8m}$;
- $l = \log \bar{n}$, где $\bar{n}$ берётся из теоремы 5.5;
- $d = O(l^2 \log m)$ берётся из теоремы 5.3.

Можно заметить, что выбор d однозначно задаёт все остальные параметры: по k и d определяется m, затем δ, затем $\bar{n}$ и l, и для корректности остаётся проверить только существование слабого дизайна. Поскольку $l = O(\log n)$, то при выборе подходящего $d = O(\log^3 n)$ в последнем уравнении левая часть станет не меньше правой, поэтому слабый дизайн существовать будет. Именно для такого d мы зафиксируем все параметры.

Чтобы определить, как по a найти p с использованием функции TR_δ, нужна некоторая подготовка. Обозначим через L_b множество всех слов, имеющих условную сложность с ограничением на время $t_1(n)$ при известном b меньше k:

$$L_b = \left\{ u \in \{0,1\}^n \mid \mathrm{C}^{t_1(n)}(u|b) < k \right\}.$$

По условию $a \in L_b$. Кроме того, размер L_b меньше 2^k. Значит, образ $L_b \times \{0,1\}^d$ под действием функции TR_δ содержит менее 2^{k+d} элементов, т.е. в силу выбора $m = k + d + 1$ занимает менее 50% области значений $\{0,1\}^m$. Воспользуемся этим следующим образом.

Пусть $\mathcal{B}\colon \{0,1\}^m \to \{0,1\}$ — предикат, означающий «лежать в образе $L_b \times \{0,1\}^d$ под действием функции TR_δ». Иными словами, $\mathcal{B}(z) = 1$, если и только если $z = \mathrm{TR}_\delta(u, y)$ для некорого u, такого что $\mathrm{C}^{t_1(n)}(u|b) < k$, и некоторого $y \in \{0,1\}^d$. Из вышеизложенного следует, что для любого $u \in L_b$ верно неравенство

$$\Pr_{y \in \{0,1\}^d}\left[\mathcal{B}(\mathrm{TR}_\delta(u,y)) = 1\right] - \Pr_{r_1,\ldots,r_m \in \{0,1\}}\left[\mathcal{B}(r_1 \ldots r_m) - 1\right] > \frac{1}{2}. \quad (5.1)$$

Действительно, первая вероятность равна 1 по определению $\mathcal{B}$, а вторая меньше $\frac{1}{2}$ в силу выбора параметров. Перепишем неравенство (5.1) через определение функции Тревисана:

$$\Pr_{y\in\{0,1\}^d}\left[\mathcal{B}\left(\hat{u}\big(y\big|_{S_1}\big)\dots\hat{u}\big(y\big|_{S_m}\big)\right)=1\right]- \\ -\Pr_{r_1,\dots,r_m\in\{0,1\}}\left[\mathcal{B}(r_1\dots r_m)-1\right]>\frac{1}{2}. \quad (5.2)$$

Далее, применим известный приём «гибридизации»: будем по очереди заменять все $\hat{u}\big(y\big|_{S_i}\big)$ на r_i. Таким образом, левую часть неравенства (5.2) можно представить в виде суммы

$$\sum_{i=1}^{m}\Big(\Pr_{y\in\{0,1\}^d,r_{i+1},\dots,r_m\in\{0,1\}}\left[\mathcal{B}\left(\hat{u}\big(y\big|_{S_1}\big)\dots\hat{u}\big(y\big|_{S_{i-1}}\big)\hat{u}\big(y\big|_{S_i}\big)r_{i+1}\dots r_m\right)=1\right]- \\ -\Pr_{y\in\{0,1\}^d,r_i,r_{i+1},\dots,r_m\in\{0,1\}}\left[\mathcal{B}\left(\hat{u}\big(y\big|_{S_1}\big)\dots\hat{u}\big(y\big|_{S_{i-1}}\big)r_ir_{i+1}\dots r_m\right)=1\right]\Big). \quad (5.3)$$

Поскольку это выражение является суммой m слагаемых и при этом превышает $\frac{1}{2}$, то хотя бы одно слагаемое превышает $\frac{1}{2m}$. Получаем, что для некоторого $i\in[1,m]$ верно

$$\Pr_{y\in\{0,1\}^d,r_{i+1},\dots,r_m\in\{0,1\}}\left[\mathcal{B}\left(\hat{u}\big(y\big|_{S_1}\big)\dots\hat{u}\big(y\big|_{S_{i-1}}\big)\hat{u}\big(y\big|_{S_i}\big)r_{i+1}\dots r_m\right)=1\right]- \\ -\Pr_{y\in\{0,1\}^d,r_i,r_{i+1},\dots,r_m\in\{0,1\}}\left[\mathcal{B}\left(\hat{u}\big(y\big|_{S_1}\big)\dots\hat{u}\big(y\big|_{S_{i-1}}\big)r_ir_{i+1}\dots r_m\right)=1\right]>\frac{1}{2m} \quad (5.4)$$

Более того, биты y вне S_i можно фиксировать некоторым образом, так чтобы неравенство (5.4) осталось верным. Обозначим слово $y\big|_{S_i}$ через x. Тогда при фиксированных битах вне x каждая функция $\hat{u}\big(y\big|_{S_j}\big)$ будет зависеть только от $|S_i\cap S_j|$ битов, лежащих в x. Обозначим эту функцию через f_j. Таким образом, для каждого j слово u задаёт булеву функцию

$$f_j\colon\{0,1\}^{|S_i\cap S_j|}\to\{0,1\}.$$

В дальнейшем мы для простоты будем записывать $\hat{u}\big(y\big|_{S_j}\big)$ как $f_j(x)$, хотя существенным образом f_j зависит только от $|S_i\cap S_j|$ битов x. Таким образом, таблица истинности для такой функции занимает $2^{|S_i\cap S_j|}$ битов, а совокупность таблиц истинности для функций $f_1,\dots,f_{i-1}$ занимает

$\sum_{j=1}^{i-1} 2^{|S_i \cap S_j|}$ битов. По определению слабого дизайна это число не больше m.

Теперь можно определить слово p для данного a. Это будет совокупность таблиц истинности для функций $f_1, \ldots, f_{i-1}$, построенных по $u = a$. Нетрудно проверить два из трёх условий теоремы. Во-первых, длина p меньше m, т.е. не больше $k + d$, т.е. не больше $k + O(\log^3 n)$, что и требовалось. Во-вторых, для задания p при известном a необходимо специфицировать k и d (которые зададут все остальные параметры, при этом n уже известно как длина a), индекс i и биты y вне x, при которых истинно неравенство 5.4. Поскольку $i \leqslant m$, а $|y| = d = O(\log^3 n)$, всё это займёт не больше $O(\log^3 n)$ битов. При этом время построения p будет полиномиальным. Действительно, по d, k и n можно посчитать m, δ и l. Затем за полиномиальное от n время строятся слабый дизайн $\{S_1, \ldots, S_m\}$ и код $\hat{a}$. Наконец, согласно переданной информации фиксируются биты вне S_i. После этого p получается переписыванием необходимых битов $\hat{a}$. Таким образом, утверждение $\mathrm{C}^{\mathrm{poly}(n)}(p|a) < O(\log^3 n)$ доказано.

Заметим также, что при известных параметрах n, k, d и i по слову p можно восстановить все таблицы истинности функций $f_1, \ldots, f_{i-1}$, несмотря на отсутствие разделителей. Действительно, по параметрам n, k и d можно восстановить дизайн $\{S_1, \ldots, S_m\}$, и при известном i вычислить размеры $|S_i \cap S_j|$ для всех $j = 1, \ldots, i-1$. После этого слово p можно разбить на блоки длины $2^{|S_i \cap S_j|}$, которые зададут таблицы функций f_j.

В следующем разделе мы докажем, что при известных p и b достаточно небольшой дополнительной информации для восстановления a полиномиальным алгоритмом из класса AM.

5.3 Доказательство корректности конструкции

В этом разделе мы докажем, что для слова p, построение которого описано в разделе 5.2, и некоторого полинома $t_2(\cdot)$ действительно выполнено соотношение $\mathrm{CAM}^{t_2(n)}(a|b,p) = O(\log^3 n)$. Начнём с преобразования неравенства (5.4) к более удобному виду. Вначале перепишем его в новых обо-

значениях.

$$\Pr_{x\in\{0,1\}^l, r_{i+1},\ldots,r_m\in\{0,1\}}\left[\mathcal{B}\left(f_1(x)\ldots f_{i-1}(x)\hat{u}(x)r_{i+1}\ldots r_m\right)=1\right]-$$
$$-\Pr_{x\in\{0,1\}^l, r_i, r_{i+1},\ldots,r_m\in\{0,1\}}\left[\mathcal{B}\left(f_1(x)\ldots f_{i-1}(x)r_i r_{i+1}\ldots r_m\right)=1\right]>\frac{1}{2m} \quad (5.5)$$

Заметим, что оно соответствует неравенству (2.3) для $z=(x,r_{i+1},\ldots,r_m)$, $\sigma=r_i$, $f(z)=f_1(x)\ldots f_{i-1}(x)r_{i+1}\ldots r_m$, $g(z)=\hat{u}(x)$, $T(a_1\ldots a_{m-1},r)=\mathcal{B}(a_1\ldots a_{i-1}ra_i\ldots a_{m-1})$. Значит, теорема 2.38 выполнено для $G_{r_i}(z)=T(z,r_i)\oplus r_i\oplus 1$, т.е. для

$$G_{r_i}(x,r_{i+1}\ldots r_m)=\begin{cases} r_i, & \text{если } \mathcal{B}(f_1(x)\ldots f_{i-1}(x)r_i r_{i+1}\ldots r_m)=1;\\ 1-r_i, & \text{иначе.}\end{cases}$$

Таким образом, применяя неравенство (2.4), выводим

$$\Pr_{x,r_i r_{i+1}\ldots r_m}\left[G_{r_i}(x,r_{i+1}\ldots r_m)=\hat{u}(x)\right]>\frac{1}{2}+\frac{1}{2m}. \quad (5.6)$$

Далее, можно зафиксировать бит r_i, так чтобы неравенство (5.6) осталось верным. Соответствующий бит включается в описание a при известных b и p. В дальнейшем без ограничения общности мы будем считать, что $r_i=1$. Введём облегчённое обозначение:

$$G(x,r)=G(x,r_{i+1}\ldots r_m)=\begin{cases} 1, & \text{если } \mathcal{B}(f_1(x)\ldots f_{i-1}(x)1r_{i+1}\ldots r_m)=1;\\ 0, & \text{иначе.}\end{cases}$$

Перепишем неравенство (5.6):

$$\Pr_{x,r_{i+1}\ldots r_m}\left[\hat{u}(x)=G(x,r_{i+1}\ldots r_m)\right]>\frac{1}{2}+\frac{1}{2m}. \quad (5.7)$$

На этом этапе наконец можно описать основную идею восстановления a по известным b и p. Зная b, можно вычислять предикат $\mathcal{B}$ на любом заданном аргументе. Зная p, можно вычислять все $f_i(x)$, и таким образом вычислять $f(z)$. Наконец, если помимо b и p известно r_i, то можно вычислять $G(x,r)$. При удачном выборе $r_{i+1}\ldots r_m$ строка $\tilde{u}=G(\cdot,r_{i+1}\ldots r_m)$ совпадёт с $\hat{u}$ более чем на доле $\frac{1}{2}+\delta$ от всех x. После этого мы применим к $\tilde{u}$ алгоритм декодирования списком и среди предложенных вариантов найдём a. К сожалению, этот план в чистом виде не сработает из-за ограничений по времени: для вычисления предиката $\mathcal{B}$ нужно, по всей видимости, сверхполиномиальное время, поскольку требуется вычислить сложность C^{poly}, а

также обратить функцию Тревисана. Зато, если значение предиката $\mathcal{B}$ равно 1, то этот факт можно удостоверить сертификатом, для чего и нужен Мерлин.

Опишем процедуру более подробно. Напомним, что слово $v' \in \{0,1\}^{\bar{n}}$ является α-аппроксимацией слова $v \in \{0,1\}^{\bar{n}}$, если v и v' совпадают хотя бы на $\alpha\bar{n}$ битах. Для фиксированного аргумента r функция $G(x,r)$ определяет некоторую булеву функцию $G^{(r)}\colon \{0,1\}^l \to \{0,1\}$. Длина таблицы истинности этой функции равна $2^l = \bar{n}$. Для фиксированного r запишем эту таблицу словом $z^{(r)} \in \{0,1\}^{\bar{n}}$. Таким образом, биты слова $z^{(r)}$ пронумерованы словами $x \in \{0,1\}^l$, и x-ый бит $z^{(r)}$ равен единице тогда и только тогда, когда $G(x,r) = 1$. Следовательно, количество единиц в слове $z^{(r)}$ равно числу слов x, для которых выполнено $\mathcal{B}(f_1(x)\dots f_{i-1}(x)1r) = 1$.

Назовём слово $v \in \{0,1\}^{\bar{n}}$ *кандидатом*, если оно является кодовым (т.е. лежит в образе $\mathrm{LDC}_{n,\delta}$) и хотя бы для доли $\frac{1}{32m}$ всех $r \in \{0,1\}^{m-i}$ соответствующая строка $z^{(r)}$ является $\left(\frac{1}{2}+\delta\right)$-аппроксимацией v.[1] Нетрудно заметить, что слово $\hat{u}$ при $u \in L_b$ является кандидатом. Действительно, предположив обратное, обозначим событие «$z^{(r)}$ является $\left(\frac{1}{2}+\delta\right)$-аппроксимацией $\hat{u}$» через $\mathcal{A}$ и распишем $\Pr_{x,r}\left[G(x,r) = \hat{u}(x)\right]$ по формуле полной вероятности:

$$\begin{aligned}&\Pr_{x,r}\left[G(x,r) = \hat{u}(x)\right] = \\ &\quad = \Pr_r\left[\mathcal{A}\right]\Pr_{x,r}\left[G(x,r) = \hat{u}(x)|\mathcal{A}\right] + \Pr_r\left[\overline{\mathcal{A}}\right]\Pr_{x,r}\left[G(x,r) = \hat{u}(x)|\overline{\mathcal{A}}\right]\end{aligned}$$

Это выражение будет ограничено сверху величиной

$$\frac{1}{32m}\cdot 1 + 1\cdot\left(\frac{1}{2}+\frac{1}{8m}\right) < \frac{1}{2}+\frac{1}{4m}, \tag{5.8}$$

что противоречит неравенству (5.7).[2] В левой части неравенства (5.8) использовано, что все вероятности не больше 1, $\Pr_r\left[\mathcal{A}\right] \leqslant \frac{1}{32m}$ по предположению, а при ложном $\mathcal{A}$ для любого фиксированного r вероятность $\Pr_x\left[G(x,r) = \hat{u}(x)\right]$ меньше $\frac{1}{2}+\delta = \frac{1}{2}+\frac{1}{8m}$ по определению $\mathcal{A}$. При этом в силу теоремы 5.5 кандидатов не может быть больше чем $32mq$, где $q = \mathrm{poly}(n/\delta) = \mathrm{poly}(n)$ — размер списка при декодировании. Действительно, для каждого r найдётся не больше q кодовых слов, которые

[1] Неявно определение кандидата зависит от исходного слова u: ведь через него определяются функции f_j, а затем предикат G.

[2] Это рассуждение осталось бы в силе, даже если заменить $\frac{1}{32m}$ на $\frac{1}{8m}$ в определении кандидата, но в дальнейшем нам понадобится именно такая оценка.

$\left(\frac{1}{2}+\delta\right)$-аппроксимируют $z^{(r)}$. Значит, общее количество пар вида (кодовое слово, $\left(\frac{1}{2}+\delta\right)$-аппроксимирующее его $z^{(r)}$), не превышает $q2^{m-i}$. Поскольку каждый кандидат $\left(\frac{1}{2}+\delta\right)$-аппроксимирует хотя бы $\frac{1}{32m}2^{m-i}$ слов вида $z^{(r)}$, то общее количество кандидатов по принципу Дирихле не превышает $q2^{m-i}/(\frac{1}{32m}2^{m-i}) = 32mq = \mathrm{poly}(n)$. Следовательно, число слов, коды которых являются кандидатами, также полиномиально, при том что код a является кандидатом. По следствию 2.12 существует программа p', работающая полиномиальная время и имеющая длину $O(\log n)$, принимающая a и отвергающая все остальные слова, коды которых являются кандидатами. Осталось составить список этих слов, для чего понадобятся случайные биты и «магия Мерлина».

Обозначим через $\bar{g}$ число $\sum_{x,r} G(x,r)/2^{m-i}$, т.е. среднее по r количество единиц в $z^{(r)}$. Заметим, что любой факт «$G(x,r)=1$» можно удостоверить сертификатом полиномиальной длины, состоящим из слов $u \in \{0,1\}^n$, $y \in \{0,1\}^d$ и программы π длины не более k со следующими свойствами:

- $\mathrm{TR}_\delta(u,y) = f_1(x)\dots f_{i-1}(x)1r$;
- $\pi(b) = u$, при этом $\pi(b)$ работает не больше $t_1(n)$ шагов.

Из второго условия следует, что $\mathrm{C}^{t_1(n)}(u|b) \leqslant k$, т.е. $u \in L_b$. Тогда из первого условия следует, что $f_1(x)\dots f_{i-1}(x)1r$ лежит в образе $L_b \times \{0,1\}^d$ под действием TR_δ. Это означает, что $\mathcal{B}(f_1(x)\dots f_{i-1}(x)1r) = 1$, т.е. $G(x,r) = 1$, что и требовалось. Ясно также, что оба условия (при известных b и p) можно проверить за полиномиальное время.

Начнём описание AM-протокола, генерирующего a. Напомним, что мы используем следующую метафору: сначала Артур получает случайные биты, затем Мерлин узнаёт эти случайные биты и посылает Артуру некоторое сообщение. Затем Артур проводит некоторые вычисления и выдаёт либо символ ошибки $\perp$, либо некоторое слово u. Требуется, чтобы с вероятностью не меньше $\frac{2}{3}$ Артур возвращал a для какого-то сообщения Мерлина, при этом для любого другого сообщения возвращал бы либо то же a, либо символ ошибки. Перейдём к собственно протоколу. Первым делом Артур выбирает равномерно и независимо случайные слова $r^{(1)},\dots,r^{(s)}$ длины $(m-i)$, где $s = s(n)$ — полином, который будет определён позже. Затем он запрашивает у Мерлина $s(\bar{g}-\gamma)$ сертификатов того, что различные пары $(x, r^{(j)})$ удовлетворяют соотношению $G(x,r^{(j)}) = 1$, где величина $\gamma = \gamma(n)$

также будет определена позже. Артур проверяет, что все пары действительно разные и что все сертификаты подходят. Если хотя бы один из них недействителен, Артур останавливается с возвращением символа ошибки. Если же все сертификаты проходят проверку, то Артур вычисляет слова $\tilde{z}_1, \ldots, \tilde{z}_s \in \{0,1\}^{\bar{n}}$ согласно следующему правилу: x-ый бит слова $\tilde{z}_j$ равен 1 тогда и только тогда, когда Мерлин предоставил сертификат того, что $G(x, r^{(j)}) = 1$. Для дальнейшего нам понадобится следующая лемма, доказанная в [13]. Помимо прочего, она специфицирует параметры s и γ.

Лемма 5.7 ([13]). *Пусть функции* $\hat{u}\colon \{0,1\}^l \to \{0,1\}$ *и* $G\colon \{0,1\}^l \times \{0,1\}^{m-i} \to \{0,1\}$ *удовлетворяют неравенству* (5.7), *а* $\bar{g} = \sum_{x,r} G(x,r)/2^{m-i}$. *Тогда для некоторого рационального* $\gamma = \bar{n}/\operatorname{poly}(m)$ *и целого* $s = \operatorname{poly}(n)$ *для равномерно и независимо выбранных слов* $r^{(1)}, \ldots, r^{(s)}$ *длины* $(m-i)$ *с вероятностью больше* $\frac{2}{3}$ *выполнены одновременно следующие условия:*

- *Хотя бы для* $s\cdot(\bar{g}-\gamma)$ *пар* $(x, r^{(j)}) \in \{0,1\}^l \times \{r^{(1)}, \ldots, r^{(s)}\}$ *выполнено условие* $G(x, r^{(j)}) = 1$;
- *При любом выборе* $s\cdot(\bar{g}-\gamma)$ *пар* $(x, r^{(j)}) \in \{0,1\}^l \times \{r^{(1)}, \ldots, r^{(s)}\}$, *удовлетворяющих условию* $G(x, r^{(j)}) = 1$, *хотя бы* $\frac{s}{16m}$ *из* s *слов* $\tilde{z}_1, \ldots, \tilde{z}_s$ *являются* $(\frac{1}{2} + \frac{1}{4m})$-*апрроксимациями* $\hat{u}$ *(где* x-*ый бит слова* $\tilde{z}_j$ *равен единице, если пара* $(x, r^{(j)})$ *находится в числе выбранных);*
- *При любом выборе* $s \cdot (\bar{g} - \gamma)$ *пар* $(x, r^{(j)}) \in \{0,1\}^l \times \{r^{(1)}, \ldots, r^{(s)}\}$, *удовлетворяющих условию* $G(x, r^{(j)}) = 1$, *любое кодовое слово* v *для* $\mathrm{LDC}_{n,\delta}$, *являющееся* $(\frac{1}{2} + \frac{1}{4m})$-*апрроксимацией хотя бы для* $\frac{s}{16m}$ *слов* $\tilde{z}_1, \ldots, \tilde{z}_s$, *является кандидатом.*

Доказательство. Значения s и γ мы выберем в конце доказательства. Заметим, что число $\bar{g}$ равняется среднему по r числу единиц в наборе $\{G(x,r)\}\big|_{x\in\{0,1\}^l}$. Если рассмотреть s таких наборов, то среднее суммарное число единиц составит $s\bar{g}$. Первое условие говорит о том, что с высокой вероятностью фактическое суммарное число единиц превысит $s(\bar{g}-\gamma)$. Формально это условие следует из неравенства Хёффдинга. Введём случайную величину ξ_i, равную доле слов x, для которых $G(x, r^{(i)}) = 1$. Тогда математическое ожидание ξ_i равно $\bar{g}/\bar{n}$ (мы поделили среднее количество нужных слов $\bar{g}$ на общее количество $\bar{n} = 2^l$). По теореме 2.39 вероятность того, что $\sum_{i=1}^{s} \xi_i < s \cdot \left(\frac{\bar{g}}{\bar{n}} - \frac{\gamma}{\bar{n}}\right)$, не превосходит $\exp(-2\gamma^2 s/\bar{n}^2)$. При этом величина

$\bar{n} \cdot \sum_{i=1}^{s} \xi_i$ как раз и равняется числу пар $(x, r^{(j)}) \in \{0,1\}^l \times \{r^{(1)}, \ldots, r^{(s)}\}$, удовлетворяющих условию $G(x, r^{(j)}) = 1$, поэтому число таких пар не меньше $s(\bar{g} - \gamma)$ с вероятностью хотя бы $\exp(-2\gamma^2 s/\bar{n}^2)$. Мы также будем считать, что количество пар $(x, r^{(j)})$, удовлетворяющих условию $G(x, r^{(i)}) = 1$, не превышает $s \cdot (\bar{g} + \gamma)$. Вероятность обратного также не превосходит $\exp(-2\gamma^2 s/\bar{n}^2)$. Мы выберем s и γ так, чтобы обе вероятности были малы.

Второе условие неформально обосновывается так: если вместо $\tilde{z}_j$ взять слова $z_j = z^{(r^{(j)})}$, построенные из битов $G(x, r^{(j)})$, то общее количество единиц во всех словах составит в среднем $s\bar{g}$, и при этом каждое из этих слов в силу неравенства (5.7) будет «в среднем» $\left(\frac{1}{2} + \frac{1}{2m}\right)$-аппроксимацией $\hat{u}$. Отсюда можно заключить аналогично выкладке (5.8), что с высокой вероятностью хотя бы $\frac{s}{4m}$ этих слов будут $\left(\frac{1}{2} + \frac{1}{4m}\right)$-аппроксимациями $\hat{u}$. Условие говорит о том, что если оставить из единиц в словах z_j только $s(\bar{g} - \gamma)$ штук, т.е. перейти к словам $\tilde{z}_j$, то всё равно с высокой вероятностью хотя бы $\frac{s}{16m}$ слов останутся $\left(\frac{1}{2} + \frac{1}{4m}\right)$-аппроксимациями $\hat{u}$. Идея заключается в том, что γ достаточно мало́, чтобы изменение в среднем $s\gamma$ битов с единицы на ноль не могло «испортить» слишком много слов z_j. Далее мы изложим эту идею формально.

Оценим вероятность невыполнения второго условия, т.е. ситуации, когда меньше $\frac{s}{16m}$ слов $\tilde{z}_1, \ldots, \tilde{z}_s$ являются $\left(\frac{1}{2} + \frac{1}{4m}\right)$-аппроксимациями $\hat{u}$. В этом случае общее количество пар $(x, r^{(j)})$, для которых выполнено $\tilde{z}_j(x) = \hat{u}(x)$, меньше, чем $\frac{s}{16m} \cdot \bar{n} + s \cdot (\frac{1}{2} + \frac{1}{4m})\bar{n} = s\bar{n} \cdot (\frac{1}{2} + \frac{5}{16m})$. Поскольку (как мы договорились) общее количество пар $(x, r^{(j)})$, для которых $G(x, r^{(j)}) = 1$, не превосходит $s(\bar{g} + \gamma)$, а общее количество единиц во всех словах $\tilde{z}_j$ не меньше $s(\bar{g} - \gamma)$, то общее количество таких пар, что $G(x, r^{(j)})$ не совпадает с $\tilde{z}_j(x)$, и потому может совпадать с $\hat{u}(x)$, не превышает $2\gamma s$. Значит, количество пар, для которых $G(x, r^{(j)}) = \hat{u}(x)$, не больше $s\bar{n} \cdot (\frac{1}{2} + \frac{5}{16m}) + 2\gamma s = s\bar{n} \cdot (\frac{1}{2} + \frac{5}{16m} + 2\frac{\gamma}{\bar{n}})$. Если $\gamma < \frac{\bar{n}}{32m}$, то эта величина меньше $s\bar{n} \cdot (\frac{1}{2} + \frac{3}{8m})$. Однако в силу неравенства (5.7) усреднённое по случайному выбору r количество x, для которых выполнено $G(x, r) = \hat{u}(x)$, равно $\bar{n}(\frac{1}{2} + \frac{1}{2m})$. Поэтому количество пар $(x, r^{(j)})$, для которых $G(x, r^{(j)}) = \hat{u}(x)$, есть сумма s случайных величин, каждая из которых имеет математическое ожидание не меньше $\bar{n}(\frac{1}{2} + \frac{1}{2m})$. Вновь применив неравенство Хёффдинга, получаем, что вероятность того, что эта сумма меньше $s\bar{n} \cdot (\frac{1}{2} + \frac{3}{8m})$, не превосходит $\exp\left(-2\frac{1}{64m^2}s\right) = \exp\left(-\frac{s}{32m^2}\right)$. Эта вероятность мала, если s существенно больше m.

Третье условие заключается в следующем: мы знаем, что кодовое слово v является $\left(\frac{1}{2}+\frac{1}{4m}\right)$-аппроксимацией хотя бы для $\frac{s}{16m}$ слов $\tilde{z}_1,\ldots,\tilde{z}_s$. Нам нужно доказать, что при рассмотрении множества всех слов $z^{(r)}$ вместо ограниченной и зашумленной выборки $\tilde{z}_j$ слово v останется $\left(\frac{1}{2}+\frac{1}{8m}\right)$-аппроксимацией хотя бы для доли $\frac{1}{32m}$ слов $z^{(r)}$. Идея доказательства вновь заключается в том, что при переходе от слов $\tilde{z}_j$ к словам z_j не больше $2s\gamma$ битов сменят своё значение с нуля на единицу. Это количество слишком мало́, чтобы слово v перестало быть хорошей аппроксимацией для слов z_j. А при достаточно большой выборке v будет хорошей аппроксимацией и для всех слов $z^{(r)}$.

Формально мы оценим вероятность невыполнения третьего условия, т.е. ситуации, когда некоторое кодовое слово v одновременно является $(\frac{1}{2}+\frac{1}{4m})$-аппроксимацией хотя бы для $\frac{s}{16m}$ слов $\tilde{z}_1,\ldots,\tilde{z}_s$ и не является $(\frac{1}{2}+\frac{1}{8m})$-аппроксимацией ни для каких $\frac{2^{m-i}}{32m}$ слов $z^{(r)}$ при $r\in\{0,1\}^{m-i}$. Поскольку слова $\tilde{z}_1,\ldots,\tilde{z}_s$ отличаются от $z_1,\ldots,z_s$ суммарно не более чем в $2s\gamma$ позициях, то при $\gamma\leqslant\frac{\bar{n}}{1024m^2}$ не может быть так, чтобы у $\frac{s}{64m}$ слов z_j доля совпадающих с v битов снизилась на $\frac{1}{8m}$ по сравнению с $\tilde{z}_j$. Значит, v является $(\frac{1}{2}+\frac{1}{8m})$-аппроксимацией хотя бы для $\frac{3s}{64m}$ слов z_j. Оценим вероятность, что при этом v не является $(\frac{1}{2}+\frac{1}{8m})$-аппроксимацией хотя бы для доли $\frac{1}{32m}$ слов $z^{(r)}$. Пусть ζ_j — случайная величина, равная 1, если z_j является $(\frac{1}{2}+\frac{1}{8m})$-аппроксимацией v, и 0 в противном случае, а $\beta<\frac{1}{32m}$ — вероятность того, что $\zeta_j=1$. Если v является $(\frac{1}{2}+\frac{1}{8m})$-аппроксимацией хотя бы для $\frac{3s}{64m}$ слов z_j, то $\sum_{j=1}^{s}\zeta_j>s(\beta+\frac{1}{64m})$. По неравенству Хёффдинга вероятность этого факта не превышает $\exp\left(-2\frac{1}{4096m^2}s\right)=\exp\left(-\frac{s}{2048m^2}\right)$. Суммирование по всем кодовым словам добавляет множитель 2^n.

Таким образом, общая вероятность того, что хотя бы одно из условий не выполнено, не превосходит

$$2\exp\left(-\frac{2\gamma^2 s}{\bar{n}^2}\right)+\exp\left(-\frac{s}{32m^2}\right)+2^n\exp\left(-\frac{s}{2048m^2}\right). \tag{5.9}$$

Кроме того, мы потребовали, чтобы $\gamma\leqslant\frac{\bar{n}}{1024m^2}$ (откуда следует и требование $\gamma\leqslant\frac{\bar{n}}{32m}$). Если положить $\gamma=\frac{\bar{n}}{1024m^2}$ и $s=n^5$, то выражение (5.9) будет меньше $\frac{1}{3}$, что и требовалось доказать в лемме. □

Вернёмся к AM-протоколу. Напомним, что Артур вычислил слова $\tilde{z}_1,\ldots,\tilde{z}_s\in\{0,1\}^{\bar{n}}$. Для каждого из слов $\tilde{z}_i$ Артур применяет алгоритм декодирования списком и получает список $\mathcal{V}_i$ всех слов $x\in\{0,1\}^n$, коды

которых являются $\left(\frac{1}{2}+\frac{1}{8m}\right)$-аппроксимациями $\tilde{z}_i$. Затем Артур объединяет списки и оставляет только те слова, которые встретились хотя бы $s/16m$ раз. По лемме 5.7 с вероятностью хотя бы $2/3$ в новом списке есть слово $a = \mathrm{LDC}^{-1}_{n,\delta}(\hat{u})$ (об этом говорит второе условие леммы), а все слова являются прообразами кандидатов (об этом говорит третье условие леммы). Значит, программой p' (существующей по следствию 2.12 и имеющей логарифмическую длину) можно отделить слово a от всех остальных, этой программой воспользуется Артур и таким образом найдёт a.

Проверим, что объём дополнительной информации, нужный для восстановления a, не превышает $O(\log^3 n)$. Действительно, помимо слов b и p Артуру нужно получить следующую информацию:

- Параметры n, k, d, i, по которым можно восстановить все остальные параметры, в том числе s и γ;
- Бит r_i, при котором неравенство (5.6) остаётся верным;
- Приближение к числу $\bar{g}$, достаточное для вычисления целой части $s \cdot (\bar{g} - \gamma)$;
- Программу p', позволяющую отличить $\hat{u}$ от остальных кандидатов.

Вся эта информация занимает даже $O(\log n)$, а не $O(\log^3 n)$, так что условие выполнено с запасом.[3]

Опишем ещё раз AM-алгоритм и докажем его корректность:

1. Артур вычисляет значения параметров m, δ, $\bar{n}$, l, s, γ по известным ему n, k, d, i;
2. Артур выбирает случайно и независимо слова $r^{(1)}, \ldots, r^{(s)} \in \{0,1\}^{m-i}$;
3. Мерлин посылает $s \cdot (\bar{g} - \gamma)$ сертификатов того, что $G(x, r^{(j)}) = 1$ для различных пар $(x, r^{(j)})$ (с указанием, какой сертификат относится к какой паре);
4. Артур проверяет, что сертификатов именно столько (используя приближения к $\bar{g}$), что все они соответствуют различным парам и что они все подходят (используя b, p и r_i). Если хотя бы один сертификат не подошёл, Артур останавливает вычисления и возвращает $\perp$;

[3] Напомним, что $O(\log^3 n)$ битов получится, если потребовать $|p| < k$ вместо $|p| < k + O(\log^3 n)$.

5. Артур вычисляет слова $\tilde{z}_1, \ldots, \tilde{z}_s \in \{0,1\}^{\bar{n}}$ на основе присланных ему сертификатов: $\tilde{z}_j(x) = 1$, если был прислан сертификат того, что $G(x, r^{(j)}) = 1$;

6. Артур декодирует слова $\tilde{z}_1, \ldots, \tilde{z}_s$ списками и выбирает те слова из списков, которые встречаются хотя бы $s/16m$ раз;

7. Ко всем выбранным словам Артур применяет программу p'. Если p' не приняла ни одного слова или приняла более одного слова, Артур останавливает вычисления и возвращает $\perp$. Если p' приняла ровно одно слово, то Артур возвращает его в качестве a.

Как нетрудно заметить, все шаги работают полиномиальное время, общая длина сертификатов также полиномиальна. Проверим, что Артур действительно вычисляет a с вероятностью хотя бы $\frac{2}{3}$. Это следует из леммы 5.7: с вероятностью хотя бы $\frac{2}{3}$ одновременно существует набор сертификатов, при которых Артур восстанавливает a, и не существует набора сертификатов, позволяющего обмануть Артура. А именно, по первому условию леммы существует набор сертификатов, который Артур принимает на стадии 4, а по второму и третьему условию любой набор сертификатов, принимаемый на этой стадии, приводит к построению списка слов, состоящего только из прообразов кандидатов и содержащему a, на стадии 6. После этого на стадии 7 из этого списка будет отобрано a, что и требовалось.

Таким образом, условие $\mathrm{CAM}^{t_2(n)}(a|b,p) = O(\log^3 n)$ доказано.

5.4 О теореме для нескольких условий

Возникает вопрос, можно ли распространить теорему Мучника для CAM-сложности на два или большее количество условий. С одной стороны, функция Тревисана является префиксным экстрактором (если взять вместо слабого дизайна равномерно слабый, см. ниже), поэтому есть потенциальная возможность использовать её для теоремы с несколькими условиями. С другой стороны, в конструкции мы не пользовались свойством экстрактора непосредственно, а вместо этого строили код специальным образом. Заранее непонятно, можно ли распространить этот способ на два условия сразу.

Мы приведём некоторые аргументы в пользу того, что непосредственно наш метод на два условия не распространяется. А именно, мы будем анализировать следующее утверждение:

Гипотеза 5.8. *Для любого полинома $t_1(n)$ существует полином $t_2(n)$, для которого выполнено следующее свойство. Пусть даны числа n, k и k', слово a длины меньше n и слова b и c, такие что $\mathrm{C}^{t_1(n)}(a|b) < k$ и $\mathrm{C}^{t_1(n)}(a|c) < k'$. Тогда существует слова p и q, одно из которых является началом другого, удовлетворяющее следующим условиям:*

- $\mathrm{CAM}^{t_2(n)}(a|b,p) = O(\log^3 n)$;
- $\mathrm{CAM}^{t_2(n)}(a|c,q) = O(\log^3 n)$;
- $|p| \leqslant k + O(\log^3 n)$;
- $|q| \leqslant k' + O(\log^3 n)$;
- $\mathrm{C}^{t_2(n)}(p|a) = O(\log^3 n)$.
- $\mathrm{C}^{t_2(n)}(q|a) = O(\log^3 n)$.

Как и в теореме с одним условием, если гипотеза верна, то некоторые из слагаемых $O(\log^3 n)$ могут быть уменьшены до $O(1)$ или $O(\log n)$, но мы оставляем их в исходном виде для единообразия.

В основной теореме длина p, т.е. сумма длин записей таблиц истинности функций f_j оценивалась как $m-1$ независимо от i. Теперь нужно, чтобы часть этих таблиц можно было взять в качестве q, которое должно быть меньшей длины, поэтому нужна более точная оценка. Естественным способом представляется рассмотрение равномерно слабых дизайнов вместо слабых:

Определение 5.9. Пусть l и d суть натуральные числа, $\rho > 1$, а $S_i \subset \{1, \ldots, d\}$ для $i = 1, \ldots, m$. Система множеств $\{S_1, \ldots, S_m\}$ называется *(l, d, ρ)-равномерно слабым дизайном*, если каждое S_i состоит из l элементов и для каждого $i > 1$ сумма $\sum\limits_{j=1}^{i-1} 2^{|S_i \cap S_j|}$ не превосходит $\rho(i-1)$.

Непосредственно из определения видно, что если система $\{S_1, \ldots, S_m\}$ является (l, d, ρ)-равномерно слабым дизайном, то и любая система $\{S_1, \ldots, S_{m'}\}$ при $m' \leqslant m$ также является (l, d, ρ)-равномерно слабым дизайном. Это ключевое свойство, благодаря которому экстрактор становится префиксным. Далее, верна следующая теорема существования:

Теорема 5.10 ([37]). *Существует алгоритм, работающий полиномиальное от $l + m$ время и генерирующий (l, d, ρ)-слабый дизайн для $d = O(l^2/\log\rho)$.*

При непосредственном применении метода нужно зафиксировать некоторый равномерно слабый дизайн, порождённый этим алгоритмом, и рассмотреть функцию Тревисана, построенную на основе этого равномерно слабого дизайна. Поскольку определение функции точно такое же, то мы сохраним для неё обозначение TR_δ. Такая функция будет являться префиксным экстрактором, но как этот факт использовать непосредственно, неясно.

По сравнению с исходной теоремой появился дополнительный параметр ρ. В силу целочисленности $\sum\limits_{j=1}^{i-1} 2^{|S_i \cap S_j|}$ неважно, равен ли ρ единице, или же $\rho = 1 + \frac{1}{m}$. А во втором случае по теореме 5.10 существует равномерно слабый дизайн с $d = O(\log^3 n)$. Остальные параметры можно выбрать аналогично теореме с одним условием.

К сожалению, дальнейшее рассуждение не обобщается на случай двух условий. Если аналог предиката $\mathcal{B}$ будет зависеть и от условия b, и от условия c, то при восстановлении a Артур не сможет вычислять его, зная только одно из условий. Если же рассмотреть два разных предиката, построенных по условиям b и c, то для них могут получиться разные места i, по которым строятся функции $f_1, \ldots, f_{i-1}$, т.е. коды p и q. В таком случае не получится, что одно из p и q будет началом другого. Более того, при использовании равномерно слабого дизайна с высокой вероятностью число i будет примерно равно m, иначе сложность $\mathrm{CAM}^{t_2(n)}(a|b)$ будет существенно меньше $\mathrm{C}^{t_1(n)}(a|b)$, что невозможно при $\mathrm{C}^{t_1(n)}(a|b) \approx \mathrm{C}(a|b)$. Аналогично для условия c число i' будет примерно равно $m' = k' + d + 1$. Таким образом, при существенном отличии k от k' числа i и i' тоже скорее всего получатся разными.

Даже если числа k и k' совпадают, всё равно для того, чтобы числа i и i' гарантированно совпали, нужно более сложное рассуждение, чем метод гибридизации. Пока что такого рассуждения найти не получается. Возможно, требуется существенно другая комбинаторная конструкция. Таким образом, аналог теоремы Мучника для CAM-сложности для двух условий пока что остаётся гипотезой.

Глава 6

Колмогоровские экстракторы с ограничением на память

В этой главе мы применим технику «наивной дерандомизации» в ситуации, существенно отличной от теоремы Мучника. А именно, мы переложим для сложности с ограничением на память теорему 2.30 о существовании обычных и усиленных колмогоровских экстракторов. В разделе 6.1 мы дадим необходимые определения и сформулируем теорему существования. В разделе 6.2 мы определим главный комбинаторный объект, использующийся в доказательстве — пёстрые и равномерно пёстрые таблицы.[1] Затем, в разделе 6.3, мы перескажем доказательство Зиманда теоремы 2.30, использующее пёстрые таблицы, и покажем, какие изменения нужно сделать для доказательства нашей теоремы. В разделе 6.4 мы опишем технические конструкции, используемые для доказательства новой теоремы, а в разделе 6.5 проведём само доказательство. Все этапы будут проводиться параллельно для обычных и усиленных колмогоровских экстракторов.

6.1 Определения и формулировки теорем

В разделе 2.3 мы определили понятия обычного и усиленного колмогоровского экстрактора, а также сформулировали результаты касательно их существования, полученные Зимандом. В этом разделе мы распространим определения на сложность с ограничением на память и сформулируем новые результаты.

Основная трудность состоит в расширении на полиномиальную память

[1] В оригинале Зиманд называет их «сбалансированными» (balanced) и «радужно сбалансированными» (rainbow balanced) таблицами. Нам представляется, что, поскольку речь пойдёт о цветах ячеек, термин «пестрота» более удачно передаёт на русском языке суть явления.

понятия зависимости двух слов. Дело в том, что величины $\mathrm{C}^s(x) - \mathrm{C}^s(x|y)$ и $\mathrm{C}^s(x) + \mathrm{C}^s(y) - \mathrm{C}^s(x, y)$ не монотонны по s. Поэтому использование подобных величин для формализации ситуации «зависимость двух слов не больше заданной величины» затруднительно. Вместо этого мы будем задавать отдельные ограничения на память для сложностей слов x, y и пары (x, y).

Определение 6.1. Пусть $s\colon \mathbb{N} \to \mathbb{N}$ — конструируемая по памяти функция, а $m\colon \mathbb{N} \to \mathbb{N}$, $k\colon \mathbb{N} \to \mathbb{N}$ и $\delta\colon \mathbb{N} \to \mathbb{N}$ — функции, вычислимые на памяти $s(n)$. Будем говорить, что семейство функций $\mathrm{KExt}_n\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^{m(n)}$ является (k, δ)-колмогоровским экстрактором с ограничением на память $s = s(n)$, если KExt_n вычислима на памяти $O(s(n))$ и для некоторой константы $\mu > 1$ для всех n и всех слов x и y длины n из условий $\mathrm{C}^s(x) > k(n)$, $\mathrm{C}^s(y) > k(n)$ и $\mathrm{C}^{\mu s}(x, y) > \mathrm{C}^s(x) + \mathrm{C}^s(y) - \delta(n)$ следует, что $\mathrm{C}^s(\mathrm{KExt}_n(x, y)) > m(n) - \delta(n) - O(\log n)$. Если, более того, для всех таких x и y выполнены условия $\mathrm{C}^s(\mathrm{KExt}_n(x, y)|x) > m(n) - \delta(n) - O(\log n)$ и $\mathrm{C}^s(\mathrm{KExt}_n(x, y)|y) > m(n) - \delta(n) - O(\log n)$, то будем называть такой экстрактор усиленным.

Надо отметить, что в работе [18] всё-таки использовалась зависимость между x и y в явном виде, но при этом для $\mathrm{C}(x)$ и $\mathrm{C}(x|y)$ были заданы разные ограничения на память. Эти ограничения соответствуют s и μs в нашем определении.

Мы докажем следующий аналог результата Зиманда 2.30:

Теорема 6.2. *Существует полином $p(n)$, такой что для любой функции $s(n) > p(n)$, конструируемой по памяти, и любых функций $1 < k(n) < n$ и $1 < \delta(n) < k(n) - O(\log n)$, вычислимых на памяти $s(n)$, существуют (k, δ)-колмогоровский экстрактор для ограничения на память $s(n)$ со значениями длины $m = 2k(n) - O(\log n)$, а также (k, δ)-колмогоровский экстрактор в сильном смысле для ограничения на память $s(n)$ со значениями длины $m = k(n) - O(\log n)$.*

6.2 Пёстрые таблицы

В этом разделе мы определим понятия пёстрых и равномерно пёстрых таблиц. Это основной комбинаторный инструмент, использованный Зиландом для доказательства теоремы 2.30. Для полноты изложения мы не толь-

ко дадим определения (которые будут слегка отличаться от зимандовских, оставаясь эквивалентными им), но и приведём доказательства существования этих объектов.

Квадратной таблицей мы будем называть функцию из $\{0,1\}^n \times \{0,1\}^n$ в $\{0,1\}^m$. При этом первый аргумент мы будем понимать как номер столбца, второй — как номер строки, а значение — как цвет соответствующей ячейки.

Определение 6.3. Назовём $(K,\ Q)$-*пёстрой таблицей* функцию $\mathrm{BT}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, обладающую следующим свойством: в любом «прямоугольнике» $S_1 \times S_2$, где $S_i \subset \{0,1\}^n$ и $|S_i| \geqslant K$, доля ячеек, покрашенных в Q самых частых в этом прямоугольнике цветов, меньше $2Q/2^m$.

Это свойство похоже на определение 2.23 экстракторов с двумя источниками при помощи раскрасок таблиц. Условие пестроты слабее свойства экстрактора при больши́х Q, а именно при $Q > \varepsilon 2^m$, и сильнее его при маленьких Q, т.е. при $Q < \varepsilon 2^m$. При этом можно заметить, что если таблица является $(K,\ Q)$-пёстрой, то она является и $(K,\ Q')$-пёстрой при $Q' > Q$. Действительно, если доля ячеек, покрашенных в Q самых частых цветов, меньше, чем $2Q/2^m$, то средняя доля ячеек этих цветов меньше $2/2^m$, а значит и доля ячеек любого оставшегося цвета меньше $2/2^m$. Таким образом, при увеличении Q на единицу доля ячеек, покрашенных в Q самых частых цветов, возрастёт меньше, чем на $2/2^m$, а верхняя оценка (т.е. $2Q/2^m$) — ровно на $2/2^m$. Следовательно, неравенство сохранится.

Также можно заметить, что свойство достаточно проверить для всех S_1 и S_2 размера в точности K. Действительно, пусть свойство нарушается, т.е. для некоторых S_1 и S_2 доля ячеек в прямоугольнике $S_1 \times S_2$, покрашенных в Q наиболее частых цветов, не меньше $2Q/2^m$. Эта доля равна средней доле ячеек, покрашенных в эти Q цветов, среди всех прямоугольников $S_1' \times S_2'$, где $S_i' \subset S_i$ и $|S_i'| = K$, $i = 1, 2$. Значит, в каком-то прямоугольнике $S_1' \times S_2'$, где $|S_i'| = K$, доля ячеек, покрашенных в эти Q цветов, не меньше $2Q/2^m$. Значит, доля ячеек, покрашенных в Q наиболее частых в этом прямоугольнике цветов, тем более не меньше $2Q/2^m$. Таким образом, если свойство нарушается для каких-то множеств, то оно нарушается и для некоторых множеств размера в точности K. Значит, достаточно потребовать его выполнения для множеств размера в точности K, что и было

заявлено.

Нам также потребуется усиленное определение, в котором самые частые цвета выбираются не во всём прямоугольнике, а в отдельных строках и столбцах.

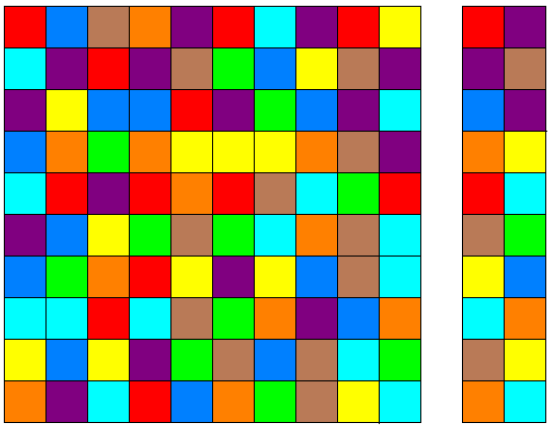
Определение 6.4. Назовём $(K,\ Q)$-*равномерно по столбцам пёстрой таблицей* функцию $\mathrm{RBT}\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, такую что для любого прямоугольника $S_1 \times S_2$, где $S_i \subset \{0,1\}^n$ и $|S_i| \geqslant K$, выполнено следующее свойство. Пусть для всех $x \in S_1$ в столбце $\{x\} \times S_2$ отмечены ячейки, покрашенные в Q наиболее частых цветов в этом столбце. Тогда доля всех отмеченных ячеек в $S_1 \times S_2$ меньше, чем $2Q/2^m$.

Аналогично определим $(K,\ Q)$-*равномерно по строкам пёструю таблицу* и назовём таблицу $(K,\ Q)$-*равномерно пёстрой*, если она равномерно пёстрая одновременно по строкам и по столбцам. Определение проиллюстрировано на рис. 6.1.

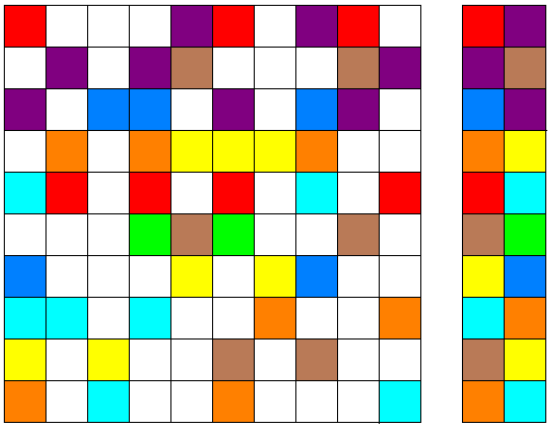
Для равномерно пёстрых таблиц верны те же замечания, что и для пёстрых. Во-первых, если таблица является $(K,\ Q)$-равномерно пёстрой, то она будет и $(K,\ Q')$-равномерно пёстрой при $Q' > Q$. Действительно, определение равномерной пестроты (по одному измерению, например, по строкам) можно понимать таким образом: пусть каждая ячейка помечена числом от 1 до 2^m, которое показывает её ранг по частоте цвета в соответствующей строке прямоугольника $S_1 \times S_2$ (самому частому цвету соответствует 1, самому редкому — 2^m, при равенстве частот приоритет отдаётся цвету с меньшим номером). Тогда чем больше номер метки, тем реже она встречается в таблице. Таблица будет равномерно пёстрой, если в любом прямоугольнике со сторонами не меньше K доля ячеек, помеченных числами от 1 до Q, меньше $2Q/2^m$. Если это так, то доля ячеек, помеченных числом Q, меньше $2/2^m$, а значит, и доля ячеек, помеченных всеми бóльшими числами, также меньше $2/2^m$. Суммируя эти доли до числа Q', получим число меньше $2Q'/2^m$, что и требовалось. Рассуждение для столбцов полностью аналогично, что и завершает рассуждение.

Во-вторых, выполнение условия достаточно потребовать только для множеств S_1 и S_2 размера в точности K. Действительно, пусть в некотором прямоугольнике $S_1 \times S_2$ доля ячеек, помеченных числами от 1 до Q, превышает $2Q/2^m$. Значит, в каком-то прямоугольнике $S_1' \times S_2'$, где $S_i' \subset S_i$ и $|S_i'| = K$, доля помеченных ячеек также превышает $2Q/2^m$. Но если те-

а)



б)



в)

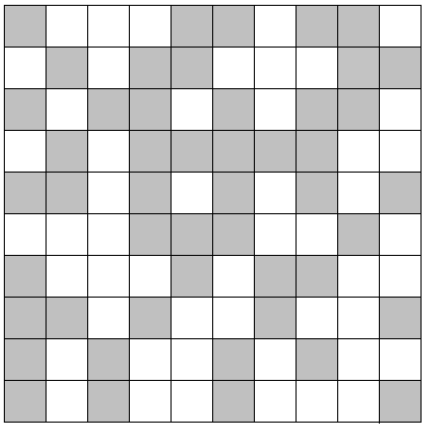


Рис. 6.1: Иллюстрация определения (10,2)-равномерно пёстрой по строкам таблицы для фиксированного прямоугольника $S_1 \times S_2$. На рисунке а) приведена раскраска квадрата размера 10×10 в $2^m = 8$ цветов и отмечены наиболее частые в строках цвета. На рисунке б) оставлена раскраска только клеток, покрашенных в эти цвета. На рисунке в) соответствующие клетки помечены серым. Всего отмечено 49 клеток, что меньше $2Q/2^m = 1/2$ от общего количества. Таким образом, условие равномерной по строкам пестроты для данного прямоугольника выполнено.

перь метки расставить не относительно $S_1 \times S_2$, а относительно $S'_1 \times S'_2$, то количество (и доля) отмеченных вершин в $S'_1 \times S'_2$ не уменьшится. Значит, в прямоугольнике $S'_1 \times S'_2$ условие также нарушится. Значит, достаточно потребовать выполнения условия только для прямоугольников размера $K \times K$, как и было заявлено.

Зиманд в работах [53; 55] доказал вероятностным методом существование пёстрых и равномерно пёстрых таблиц с определёнными параметрами. В силу того, что наше определение несколько отличается от оригинального, в том числе выбором параметров, мы приведём теоремы существования вместе с доказательствами.

Теорема 6.5. *При всех достаточно больших n и m, и всех $K \geqslant \max\{m,n\}\sqrt{2^m}$ и $Q \geqslant \max\{m,n\}\sqrt{2^m}$ существует (K, Q)-пёстрая таблица. Также для всех $K \geqslant (\max\{m,n\})^2 2^m$ и любого Q существует (K, Q)-равномерно пёстрая таблица.*

Доказательство. В силу сделанных выше замечаний достаточно доказать теорему для $K = Q = \max\{n,m\}\sqrt{2^m}$ в случае пёстрых таблиц и $K = (\max\{m,n\})^2 2^m$ и $Q = 1$ в случае равномерно пёстрых.

Доказательство строится стандартным применением вероятностного метода. Возьмём случайную таблицу данного размера и докажем, что она с положительной вероятностью является (равномерно) пёстрой. Под случайностью таблицы будем понимать, что цвета всех её ячеек выбраны равномерно и независимо друг от друга. Проведём отдельные рассуждения для пёстрых и равномерно пёстрых таблиц.

Оценим вероятность того, что таблица не является пёстрой. Это означает, что для некоторых S_1 и S_2 размера K доля ячеек, покрашенных в Q самых частых цветов, не меньше $2Q/2^m$. Значит, найдутся Q цветов и $2QK^2/2^m$ ячеек, покрашенных в эти цвета. Вероятность этого события не превышает $\left(C_{2^n}^K\right)^2 \cdot C_{2^m}^Q \cdot \Pr\left[\text{сумма } K^2 \text{ независимых бернуллиевских случайных величин со средним } \frac{Q}{2^m} \text{ больше, чем } \frac{2QK^2}{2^m}\right]$. В этом произведении $\left(C_{2^n}^K\right)^2$ — число способов выбрать S_1 и S_2, $C_{2^m}^Q$ — число способов выбрать Q цветов, а упомянутая бернуллиевская случайная величина равна единице, если данная ячейка покрашена в один из Q выбранных цветов. В силу неравенства (2.5) величина $\left(C_{2^n}^K\right)^2$ меньше $\left(\frac{e2^n}{K}\right)^{2K}$, а величина $C_{2^m}^Q$ меньше $\left(\frac{e2^m}{Q}\right)^Q$. Далее, в силу следствия 2.41 третий множитель не превышает $e^{-\frac{1}{3}\frac{Q}{2^m}K^2}$. Таким образом, вероятность того, что таблица не пёстрая, не

превышает $\left(\frac{e2^n}{K}\right)^{2K} \cdot \left(\frac{e2^m}{Q}\right)^{Q} \cdot e^{-\frac{1}{3}\frac{Q}{2^m}K^2} = e^{2K(1+n\ln 2-\ln K)+Q(1+m\ln 2-\ln Q)-\frac{1}{3}\frac{QK^2}{2^m}}$. Если взять $K = Q = \max\{m,n\}\sqrt{2^m}$, то положительные слагаемые в показателе будут иметь порядок $(\max\{m,n\})^2\sqrt{2^m}$, а последнее отрицательное слагаемое будет иметь порядок $(\max\{m,n\})^3\sqrt{2^m}$. Таким образом, при достаточно больших m и n показатель будет отрицательным, а вся величина будет меньше единицы. Значит, вероятность того, что таблица не пёстрая, меньше единицы, поэтому пёстрые таблицы существуют.

Теперь докажем существование равномерно пёстрых таблиц с указанными параметрами. Для этого докажем, что вероятность того, что случайная таблица заданного размера не является (K, Q)-равномерно по строкам пёстрой, меньше $1/2$. Аналогично получим такую же оценку для столбцов, а значит, случайная таблица будет равномерно пёстрой с положительной вероятностью. Если данная таблица не является (K, Q)-равномерно по строкам пёстрой, то найдутся S_1 и S_2 размера в точности K, доля помеченных точек в которых будет превышать $2Q/2^m$. Это значит, что в каждой строке найдутся Q выделенных цветов, такие что $2QK^2/2^m$ ячеек покрашены в выделенные цвета для соответствующих строк. Вероятность этого события не превышает $\left(C_{2^n}^K\right)^2 \cdot \left(C_{2^m}^Q\right)^K \cdot \Pr\left[\text{сумма } K^2 \text{ независимых бернуллиевских случайных величин со средним } \frac{Q}{2^m} \text{ больше, чем } \frac{2QK^2}{2^m}\right]$, что отличается от оценки для пёстрых таблиц лишь заменой $C_{2^m}^Q$ на $\left(C_{2^m}^Q\right)^K$, поскольку теперь свои Q цветов выбираются для каждой из K строк. Используя те же неравенства, что и раньше, мы получим верхнюю оценку $\left(\frac{e2^n}{K}\right)^{2K} \cdot \left(\frac{e2^m}{Q}\right)^{KQ} \cdot e^{-\frac{1}{3}\frac{Q}{2^m}K^2} = e^{2K(1+n\ln 2-\ln K)+KQ(1+m\ln 2-\ln Q)-\frac{1}{3}\frac{QK^2}{2^m}}$. Если взять $K = (\max\{m,n\})^2 2^m$ и $Q = 1$, то положительные слагаемые будут иметь порядок $(\max\{m,n\})^3 2^m$, а отрицательные — порядок $(\max\{m,n\})^4 2^m$. Таким образом, при достаточно больших n и m показатель будет меньше $-\ln 2$, а вся величина будет меньше $1/2$, что и требовалось. Значит, равномерно пёстрые таблицы с указанными параметрами существуют.

На этом теорема существования пёстрых и равномерно пёстрых таблиц доказана. □

Замечание 6.6. Заметим, что из доказательства видно, что при указанных параметрах и достаточно больших m и n (равномерно) пёстрые таблицы не только существуют, но и занимают отделённую от нуля (и даже

стремящуюся к единице) долю среди всех таблиц. Это позволит впоследствии применить лемму 2.37.

6.3 Идея доказательства основной теоремы

В этом разделе мы перескажем доказательство теоремы 2.30, принадлежащее Зиманду, и покажем, где его необходимо дополнить для доказательства теоремы 6.2.

Фактически мы покажем, что некоторая пёстрая таблица T является колмогоровским экстрактором, а некоторая равномерно пёстрая таблица — усиленным колмогоровским экстрактором. Впоследствии, чтобы доказать существование колмогоровских экстракторов с ограничением на память, мы ослабим определение пёстрой таблицы, так чтобы таблица по-прежнему была нужным экстрактором, но при этом находилась бы на ограниченной памяти.

Опишем схему доказательства для колмогоровских экстракторов. Доказательство будет строиться от противного: мы получим противоречие между сложностью пары (x, y) и простотой значения $T(x, y)$. Действительно, в силу пестроты таблицы в ней немного ячеек покрашены в цвета малой сложности, а значит, и сами эти ячейки будут иметь небольшую сложность. Опишем конструкцию подробнее, указав параметры. Напомним, что по условию сложность каждого из слов x и y не меньше k, а зависимость между ними не больше δ. Положим $d = \delta + c \log n$, где константа c будет определена позднее, и рассмотрим $(2^k, 2^{m-d})$-пёструю таблицу $T\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ (оценкой параметров займёмся позже, пока что предположим, что такая таблица существует). Определим множества $B_x = \{z \in \{0,1\}^n \mid \mathrm{C}^s(z) \leqslant \mathrm{C}^s(x)\}$, $B_y = \{z \in \{0,1\}^n \mid \mathrm{C}^s(z) \leqslant \mathrm{C}^s(y)\}$ и $A = \{w \in \{0,1\}^m \mid \mathrm{C}^s(w) < m - d\}$. Очевидно, $(x, y) \in B_x \times B_y$. Расширим произвольным образом множества B_x и B_y до множеств размеров $2^{\mathrm{C}^s(x)+1}$ и $2^{\mathrm{C}^s(y)+1}$ соответственно. Теперь к расширенному множеству $B_x \times B_y$ можно применить свойство пестроты и получить, что менее

$$2 \cdot 2^{\mathrm{C}^s(x)+1} 2^{\mathrm{C}^s(y)+1} / 2^d \tag{6.1}$$

ячеек из этого множества покрашены в цвет из множества A (действительно, меньше такого количества ячеек покрашено в 2^{m-d} самых частых цветов, потому ещё меньше будет покрашено в $|A| < 2^{m-d}$ произвольных цве-

тов). Значит, для исходного множества $B_x \times B_y$ будет верна та же оценка (по абсолютной величине, а не доле). Значит, ячейка (x, y), покрашенная в цвет из множества A, может быть описана таблицей T, множествами B_x, B_y и A, а также порядковым номером ячейки (x, y) среди всех ячеек прямоугольника $B_x \times B_y$, покрашенных в цвета из множества A. В силу оценки (6.1) последний порядковый номер требует не более $\mathrm{C}^s(x) + \mathrm{C}^s(y) - d + 3$ битов, а для описания множеств B_x, B_y и A достаточно указать n, m, d, $\mathrm{C}^s(x)$ и $\mathrm{C}^s(y)$, т.е. $O(\log n)$ битов. При этом по этим данным множества могут быть перечислены на памяти $O(s)$. Если бы существовала пёстрая таблица T сложности $O(\log n)$, вычислимая на памяти $O(s)$, то мы бы получили, что $\mathrm{C}^{O(s)}(x, y) < \mathrm{C}^s(x) + \mathrm{C}^s(y) - d + O(\log n)$, что при надлежащем выборе μ и c вступило бы в противоречие с условием $\mathrm{C}^{\mu s} > \mathrm{C}^s(x) + \mathrm{C}^s(y) - \delta$. К сожалению, такой таблицы найти не удаётся. Вместо этого мы в разделе 6.4 найдём таблицу, которая вычислима на памяти $O(s)$ и обладает некоторым более слабым свойством, достаточным для доказательства теоремы, которое мы приведём в разделе 6.5.

Теперь опишем схему доказательства для усиленных колмогоровских экстракторов. Здесь идея доказательства такова: в силу равномерной пестроты, в любом прямоугольнике будет немного ячеек с цветами малой сложности условно на номер столбца (строки), в котором (-ой) они лежат. Значит, все такие ячейки будут иметь небольшую сложность, что войдёт в противоречие с тем, что пара (x, y) сложна. Более подробно, мы возьмём равномерно пёструю таблицу с параметрами $(2^k, Q = 2^{m-d})$, где вновь $d = \delta + c \log n$ (разумеется, теперь таблица будет существовать при иных значениях параметров, но и в утверждении теоремы параметры тоже другие; точной оценкой параметров займёмся позже). Определим множества B_x и B_y так же, как и раньше. Кроме того, для каждого слова v рассмотрим множество $A_v = \{w \mid \mathrm{C}^s(w|v) < m - d\}$. Ясно, что оно содержит меньше Q элементов. Назовём слово v *плохим*, если доля ячеек в столбце $\{v\} \times B_y$, покрашенных в цвет из A_v, превышает $2 \cdot 2^{-d}$. Доля ячеек, покрашенных в Q самых частых цветов, будет не меньше, поэтому плохих столбцов будет не больше K, иначе в прямоугольнике $\{\text{плохие столбцы}\} \times B_y$ будет нарушено условие равномерной пестроты по столбцам. Если бы плохие столбцы можно было перечислить на памяти s, то каждый плохой столбец имел бы сложность не больше k, и потому x не было бы плохим столбцом. Если дополнительно предположить, что некоторая равномерно пёстрая таблица

T имеет сложность $O(\log n)$ и вычислима на памяти $O(s)$, то можно свести к противоречию предположение о том, что $T(x,y) \in A_x$. Действительно, в этом случае при известном x можно описать y следующими данными: таблицей T, множествами B_y и A_x, а также порядковым номером ячейки (x,y) среди всех ячеек в столбце $\{x\} \times B_y$, покрашенных в цвет из множества A_x. Поскольку x не плохой, то этот порядковый номер записывается не более чем $\mathrm{C}^s(y) - d + 1$ битами. Множества B_y и A_x при известном x полностью описываются параметрами n, m, d и $\mathrm{C}^s(y)$, причём их можно перечислить на памяти $O(s)$. Таким образом, сложность $\mathrm{C}^{O(s)}(y|x)$ будет меньше $\mathrm{C}^s(y) - d + O(\log n)$, откуда $\mathrm{C}^{O(s)}(x,y) < \mathrm{C}^s(x) + \mathrm{C}^s(y) - d + O(\log n)$, что противоречит условию $\mathrm{C}^{\mu s}(x,y) > \mathrm{C}^s(x) + \mathrm{C}^s(y) - \delta$ при достаточно больших c и μ. К сожалению, найти требуемую таблицу также не удаётся, поэтому мы докажем теорему с использованием более слабого свойства.

Наконец, оценим параметры колмогоровских экстракторов, которые получатся из этих конструкций. Для обычных колмогоровских экстракторов в силу теоремы 6.5 получаем требование $2^k \geqslant \max\{n,m\}2^{m/2}$ и $2^{m-d} \geqslant \max\{n,m\}2^{m/2}$. Из первого соотношения получаем $m \leqslant 2k - \log n - O(1)$, а из второго $d \leqslant k - \frac{1}{2}\log n - O(1)$, откуда $\delta \leqslant k - O(\log n)$, что и требуется в условии теоремы 6.2. Для усиленных колмогоровских эстракторов в силу теоремы 6.5 получаем требование $2^k \geqslant (\max\{m,n\})^2 2^m$, откуда $m \leqslant k - 2\log n - O(1)$. Поскольку $m - d \geqslant 0$, получаем, что $d \leqslant k - 2\log n - O(1)$, откуда $\delta \leqslant k - O(\log n)$, что также соответствует условию теоремы 6.2.

6.4 Применение генератора Нисана–Вигдерсона

В этом разделе мы опишем ослабление свойств пестроты и равномерной пестроты и покажем, что таблицы с этими свойствами встречаются среди значений генератора Нисана–Вигдерсона.

Идея, на которой основано применение генератора, та же, что и в разделах 4.2 и 4.3.2: мы сформулируем некоторое свойство, следующее из (равномерной) пестроты таблицы и при этом распознающееся схемами из функциональных элементов константной глубины. Применив лемму 2.37, мы получим, что среди значений генератора Нисана–Вигдерсона встречаются таблицы с этим свойством. Начнём с формулировки этого свойства, предварительно дав вспомогательное определение.

Определение 6.7. Пусть заданы натуральные числа $k < n$ и $q < m$. С этого момента и до конца раздела будем называть *k-релевантными* системы пар вида (S, l), где S — подмножество $\{0,1\}^n$, а l — целое число из отрезка $[k, n]$, такие что в любой паре множество S содержит не больше 2^l элементов, а вся система содержит $2^{\mathrm{poly}(n)}$ пар.[2] Также будем называть *q-релевантными* подмножества $\{0,1\}^m$, содержащие не больше 2^q элементов.

Определение 6.8. Пусть b — положительное действительное число, а $k < n$ и $q < m$ суть натуральные числа. Пусть заданы k-релевантная система пар $\mathcal{S}$ и q-релевантное множество Q. Будем говорить, что таблица T (т.е. функция $T\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$) является $(k, q, b, Q, \mathcal{S})$-пёстрой, если для любых (S_1, l_1) и (S_2, l_2) из $\mathcal{S}$ количество ячеек в $S_1 \times S_2$, покрашенных в цвета из Q, меньше $2^{l_1+l_2+q-m+b}$.

Параметр b не несёт содержательной нагрузки и введён по техническим причинам: он нужен для доказательства того, что таблицы с этим свойством встретятся среди значений генератора. В наших утверждениях он будет примерно равен 1. Заметим, что при росте b определение ослабляется, т.е. всё больше таблиц становятся $(k, q, b, Q, \mathcal{S})$-пёстрыми.

Нетрудно проверить справедливость следующего утверждения:

Утверждение 6.9. *Произвольная $(2^k, 2^q)$-пёстрая таблица является $(k, q, 1, Q, \mathcal{S})$-пёстрой при любом выборе релевантных Q и $\mathcal{S}$.*

Доказательство. Рассмотрим крайний случай: размер S_1 в точности равен 2^{l_1}, размер S_2 в точности равен 2^{l_2}, размер Q в точности равен 2^q и при этом цвета из множества Q встречаются в $S_1 \times S_2$ чаще остальных цветов. В этом случае доля ячеек из $S_1 \times S_2$, покрашенных в цвета из Q, не превосходит $2 \cdot \frac{2^q}{2^m} = 2^{1+q-m}$, откуда количество таких ячеек не превосходит $2^{l_1} \cdot 2^{l_2} \cdot 2^{1+q-m} = 2^{l_1+l_2+q-m+1}$. В остальных случаях размеры множеств S_1, S_2 или Q меньше предельных или цвета из Q не самые частые, поэтому количество таких ячеек будет ещё меньше, значит требуемая в определении 6.8 оценка будет выполнена. □

Теперь сформулируем вариант для равномерно пёстрых таблиц.

[2] В доказательстве основной теоремы такие большие системы не понадобятся, будет достаточно всего лишь системы размера $O(n)$, но вспомогательные факты верны и для таких больших систем, а уменьшение размера не даст никакого выигрыша в основной теореме.

Определение 6.10. Пусть заданы натуральные числа $k < n$ и $q < m$. С этого момента и до конца раздела будем называть *(k,q)-релевантными* системы кортежей вида $(Q_1, \ldots, Q_{2^l})$, такие что l — целое число из отрезка $[k, n]$, каждое Q_i каждого кортежа содержит менее 2^q элементов, а вся система содержит $2^{\mathrm{poly}(n)}$ кортежей.[3]

Определение 6.11. Пусть b, k и q такие же, как в определении 6.8. Пусть также заданы k-релевантная система пар $\mathcal{S}$ и (k,q)-релевантная система кортежей $\mathcal{R}$. Для фиксированных (S_1, l_1) и (S_2, l_2) из $\mathcal{S}$, а также $\mathbf{Q} = (Q_1, \ldots, Q_{2^{l_1}}) \in \mathcal{R}$ определим процедуру пометки ячеек $S_1 \times S_2$: для каждого $i \in \left[1, |S_1|\right]$ пометим ячейки из столбца $\{v_i\} \times S_2$, покрашенные в цвет из Q_i (здесь $S_1 = \{v_1, \ldots, v_{|S_1|}\}$). Назовём столбец *насыщенным*, если он содержит более $2^{l_2+q-m+b}$ помеченных ячеек. Наконец, назовём таблицу $(k, q, b, \mathcal{R}, \mathcal{S})$-равномерно по столбцам пёстрой, если для любых (S_1, l_1) и (S_2, l_2) из $\mathcal{S}$ и $\mathbf{Q} = (Q_1, \ldots, Q_{2^{l_1}}) \in \mathcal{R}$ общее количество помеченных клеток в насыщенных столбцах $S_1 \times S_2$ не больше $2^{l_2+q-m+k+b}$. Аналогично определим $(k, q, b, \mathcal{R}, \mathcal{S})$-равномерно по строкам пёструю таблицу и будем говорить, что таблица $(k, q, b, \mathcal{R}, \mathcal{S})$-равномерно пёстрая, если она равномерно пёстрая одновременно по строкам и по столбцам.

Замечание 6.12. Из определения 6.11 следует, что общее количество насыщенных строк или столбцов не больше $2^{l_2+q-m+k+b}/2^{l_2+q-m+b} = 2^k$.

Аналогично утверждению 6.9 докажем следующее

Утверждение 6.13. *Произвольная $(2^k, 2^q)$-равномерно пёстрая таблица является $(k, q, 1, \mathcal{R}, \mathcal{S})$-равномерно пёстрой при любом выборе релевантных $\mathcal{R}$ и $\mathcal{S}$.*

Доказательство. Проведём рассуждение только для равномерно по столбцам пёстрых таблиц, для равномерно по строкам пёстрых рассуждение будет аналогичным.

Пусть таблица T является $(2^k, 2^q)$-равномерно по столбцам пёстрой. Пусть размер S_1 в точности равен 2^{l_1}, а в качестве Q_i выбраны 2^q наиболее частых цветов в столбце $\{v_i\} \times S_2$. Поскольку $b = 1$, то насыщенными будут столбцы, содержащие более $2 \cdot 2^{l_2+q-m}$ отмеченных ячеек. Рассмотрим некоторое подмножество $S_1' \subset S_1$, такое что $|S_1'| = 2^k$ и либо все его

[3] Здесь также в приложении будет достаточно существенно меньших систем.

элементы являются насыщенными столбцами, либо оно содержит все насыщенные столбцы в S_1. В силу равномерной пестроты доля помеченных вершин в прямоугольнике $S_1' \times S_2$ меньше $2 \cdot 2^q / 2^m = 2^{1+q-m}$, откуда число таких вершин меньше $2^k \cdot 2^{l_2} \cdot 2^{1+q-m} = 2^{l_2+q-m+k+1}$. Отсюда число насыщенных столбцов меньше $2^{l_2+q-m+k+1} / 2^{l_2+q-m+1} = 2^k$. Значит, все насыщенные столбцы содержатся в множестве S_1', а потому число отмеченных вершин в насыщенных столбцах меньше $2^{l_2+q-m+k+1}$, что и требовалось. □

Теперь аналогично разделу 4.2.2 построим схему константной глубины, распознающую модифицированные свойства пестроты. На вход схеме будет подаваться полная таблица, т.е. размер входа равен $m2^{2n}$. Чтобы аргумент генератора оставался полиномиальным, нужно, чтобы размер схемы был экспоненциальным, т.е. $2^{\mathrm{poly}(n)}$. Такой схемы достаточно для доказательства существования нужной таблицы среди значений генератора. Действительно, по теореме 6.5 с учётом замечания 6.6 произвольная таблица с выбранными параметрами является (равномерно) пёстрой с положительной отделённой от нуля вероятностью. В силу утверждений 6.9 и 6.13 (равномерно) пёстрые таблицы являются таковыми и в смысле определений 6.8 и 6.11. Значит, произвольная таблица является (равномерно) пёстрой в смысле этих определений также с положительной отделённой от нуля вероятностью. Если мы докажем, что модифицированная (равномерная) пестрота распознаётся схемой константной глубины и размера $2^{\mathrm{poly}(n)}$, то по лемме 2.37 случайная таблица, полученная при помощи генератора Нисана–Вигдерсона, также является (равномерно) пёстрой в модифицированном смысле с положительной вероятностью. Таким образом, нужная таблица среди значений генератора будет существовать, что и требуется. К сожалению, в чистом виде этот план не срабатывает, поэтому мы докажем чуть более слабое утверждение, где будет использован параметр b.

Теорема 6.14. *Пусть заданы натуральные числа $k < n$ и $q < m$. Пусть также заданы k-релевантная система пар $\mathcal{S}$, q-релеватное множество Q и (k, q)-релеватная система кортежей $\mathcal{R}$. Тогда существуют схемы из функциональных элементов $\mathcal{BT}$ и $\mathcal{RBT}$ с $m2^{2n}$ входами и одним выходом, константной глубины и размера $2^{\mathrm{poly}(n)}$, такие что:*

- *Схема $\mathcal{BT}$ (соотв., $\mathcal{RBT}$) принимает код любой $(k, q, 1, Q, \mathcal{S})$-пёстрой (соотв., $(k, q, 1, \mathcal{R}, \mathcal{S})$-равномерно пёстрой) таблицы;*

- *Если схема $\mathcal{BT}$ (соотв., $\mathcal{RBT}$) принимает код некоторой таблицы, то эта таблица является $(k, q, 1.01, Q, \mathcal{S})$-пёстрой (соотв., $(k, q, 1.01, \mathcal{R}, \mathcal{S})$- равномерно пёстрой).*

Доказательство. Построим указанные схемы непосредственно. Они не обязаны быть равномерными, поэтому множество Q и системы $\mathcal{S}$ и $\mathcal{R}$ можно «запаять» в схему. То есть, можно построить схему для конкретного набора (S_1, S_2, l_1, l_2, Q), где $(S_1, l_1), (S_2, l_2) \in \mathcal{S}$, (или набора $(S_1, S_2, l_1, l_2, \mathbf{Q})$, где $\mathbf{Q} \in \mathcal{R}$) и взять конъюнкцию $2^{\mathrm{poly}(n)}$ копий этой схемы для разных наборов. Опишем схему для конкретного набора.

Легко проверить, является ли данная ячейка помеченной. Достаточно сравнить её цвет с каждым цветом из Q (или Q_i для строки i) и взять дизъюнкцию полученных значений. Сложнее посчитать количество помеченных ячеек и сравнить это число с порогом. Такой подсчёт невозможно выполнить точно, но возможно сделать приближённо, что будет достаточно для наших целей.

Мы будем использовать схему, существующую по теореме 2.32. Для пёстрых таблиц мы используем схему с $|S_1| \cdot |S_2|$ аргументами, которая принимает свой вход, если число единиц среди аргументов меньше $2^{l_1+l_2+q-m+1}$, и отвергает свой вход, если число единиц больше $2^{l_1+l_2+q-m+1.01}$ (если число единиц промежуточное, то схема может как принять, так и отвергнуть вход). Для равномерно пёстрых таблиц мы используем две подобных схемы последовательно. Первая схема имеет $|S_2|$ аргументов, принимает свой вход, если среди аргументов больше $2^{l_2+q-m+1.01}$ единиц, и отвергает свой вход, если среди аргументов меньше $2^{l_2+q-m+1}$ единиц. Копии этой схемы мы применяем к меткам в каждом столбце $\{v_i\} \times S_2$. Далее, нужно подсчитать суммарное число единиц в столбцах, где первая схема приняла свой вход. Для этого мы берём конъюнкции аргументов предыдущей схемы с её значениями в соответствующих столбцах. Затем к получившимся $|S_1| \cdot |S_2|$ значениям мы применим схему, которая принимает свой вход, если среди её аргументов меньше $2^{l_1+q-m+k+1}$ единиц, и отвергает его, если среди её аргументов больше $2^{l_1+q-m+k+1.01}$ единиц. Параллельно всю конструкцию нужно повторить для строк и взять конъюнкцию двух результатов.

Опишем конструкцию ещё раз, так чтобы из описания было ясно, что глубина схемы есть константа, а размер равен $2^{\mathrm{poly}(n)}$. Вначале опишем конструкцию схемы, распознающей пестроту. Аргументы схемы задают цвета всех 2^{2n} ячеек в таблице. Кроме того, добавим в схему константы для все-

возможных цветов из Q. Схема состоит из $2^{\mathrm{poly}(n)}$ одинаково устроенных блоков. Каждый блок имеет две группы входов. Первая группа задаёт цвета всех ячеек в конкретном прямоугольнике $S_1 \times S_2$. Вторая группа задаёт набор цветов Q. Разные блоки подключаются к разным входам схемы (первая группа входов блока) и к одному и тому же набору цветов Q. Каждый блок состоит из трёх уровней. На первом уровне к каждой паре цветов из первой и из второй группы применяется схема, проверяющая, совпадают ли два цвета. При простейшей реализации такая схема имеет размер $O(m)$ и глубину 4. А именно, мы запишем схемой формулу $f(\sigma_1, \ldots, \sigma_m; \tau_1, \ldots, \tau_m) = \bigwedge_{i=1}^{m}((\sigma_i \wedge \tau_i) \vee (\neg\sigma_i \wedge \neg\tau_i))$. Всего таких схем будет не больше $2^{l_1+l_2+q}$, поскольку $|S_1 \times S_2| \leqslant 2^{l_1+l_2}$, а $|Q| \leqslant 2^q$. Поскольку все три параметра l_1, l_2, q не превосходят n, то общий размер первого уровня составит $O(m2^{3n}) = 2^{\mathrm{poly}(n)}$. А поскольку все схемы на первом уровне применяются параллельно, общая глубина останется равной 4. На втором уровне для каждого элемента $S_1 \times S_2$ берётся дизъюнкция всех $|Q|$ результатов первого уровня. Это оставляет размер полиномиальным, а глубина увеличивается на 1. На третьем уровне ко всем результатам второго уровня применяется схема для приближённого подсчёта числа единиц. Как уже было сказано, эта схема принимает свой вход, если число единиц среди аргументов меньше $2^{l_1+l_2+q-m+1}$, и отвергает его, если число единиц больше $2^{l_1+l_2+q-m+1.01}$. Эта схема имеет константную глубину и размер, полиномиальный от $|S_1| \cdot |S_2| \cdot |Q|$, т.е. составляющий $2^{\mathrm{poly}(n)}$. Значит, блок в целом также имеет константную глубину и размер $2^{\mathrm{poly}(n)}$, что и требовалось. Поскольку всего блоков тоже $2^{\mathrm{poly}(n)}$, все они применяются параллельно, а на последнем этапе берётся просто конъюнкция результатов всех блоков, то и размер всей схемы будет составлять $2^{\mathrm{poly}(n)}$, а глубина её будет некоторой константой. Эскиз схемы показан на рис. 6.2.

Докажем корректность построенной схемы. Пусть ей на вход подан код $(k, q, 1, Q, \mathcal{S})$-пёстрой таблицы. Тогда в каждом множестве $S_1 \times S_2$ меньше $2^{l_1+l_2+q-m+1}$ ячеек, покрашенных в цвета из Q. Только для ячеек, покрашенных в такие цвета, хотя бы одна из $\equiv$-схем на первом выдаст 1, а значит только для таких ячеек дизъюнкция на втором уровне выдаст 1. Раз таких ячеек меньше $2^{l_1+l_2+q-m+1}$, то пороговая схема на третьем уровне выдаст 1. Поскольку такое происходит для всех прямоугольников $S_1 \times S_2$, то и вся схема выдаст 1.

Теперь пусть, напротив, схема выдала 1. Значит, для всех прямоугольни-

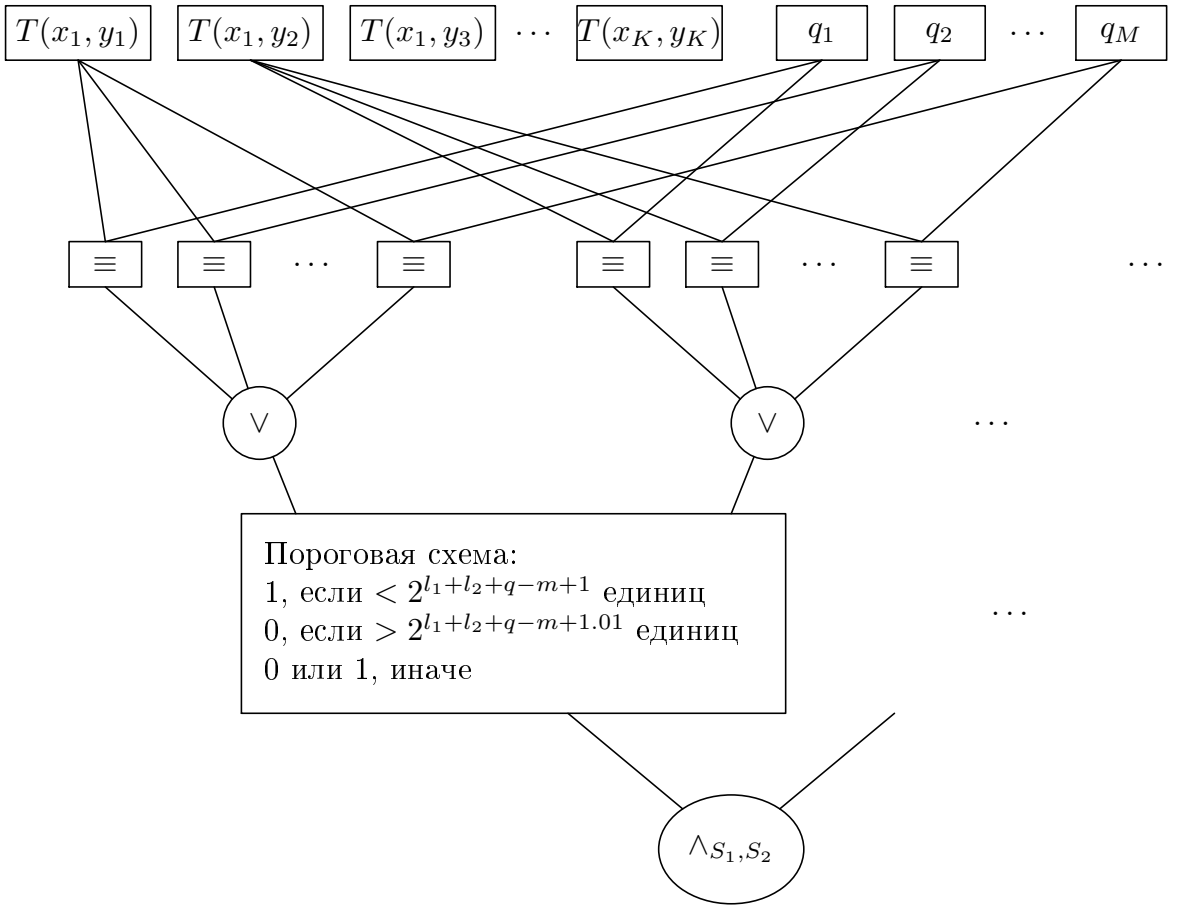


Рис. 6.2: Эскиз схемы, проверяющей пестроту таблицы.

ков $S_1 \times S_2$ соответствующая схема выдала 1. Значит, все пороговые схемы насчитали не больше $2^{l_1+l_2+q-m+1.01}$ единиц на втором уровне. Но это и значит, что в каждом $S_1 \times S_2$ не больше $2^{l_1+l_2+q-m+1.01}$ ячеек покрашены в цвета из Q, что и означает $(k, q, 1.01, Q, \mathcal{S})$-пестроту таблицы.

Перейдём к описанию конструкции для равномерно пёстрых таблиц. Поскольку теперь палитр много, добавим в схему константы для всех возможных цветов из $\{0,1\}^m$. Схема представляет собой конъюнкцию двух подсхем, устанавливающих равномерную пестроту по строкам и по столбцам. Поскольку эти подсхемы симметричны, опишем только подсхему для строк. Она также разбивается на $2^{\mathrm{poly}(n)}$ одинаково устроенных блоков, применяющихся параллельно. Каждый блок состоит из пяти уровней и применяется к цветам ячеек прямоугольника $S_1 \times S_2$ и набору палитр $\mathbf{Q}$. Первый уровень проверяет совпадение пар цветов: соответствующая схема применяется к каждой паре вида (цвет ячейки из строки $S_1 \times \{y_i\}$, цвет из Q_i). На втором уровне для каждого элемента $S_1 \times S_2$ берётся дизъюнкция всех

$|Q_i|$ результатов, соответствующих этому элементу. На третьем уровне для каждого i применяется схема для приближённого подсчёта числа единиц на втором уровне, относящихся к строке $S_1 \times \{y_i\}$. Эта схема принимает свой вход, если число единиц в нём превышает $2^{l_1+q-m+1.01}$, и отвергает его, если число единиц в нём меньше $2^{l_1+q-m+1}$. На четвёртом уровне берутся конъюнкции результатов второго и третьего уровней (единица будет означать, что ячейка помечена и строка насыщена). На пятом уровне применяется схема для приближённого подсчёта числа единиц на четвёртом уровне: она принимает вход, если он содержит меньше $2^{l_2+q-m+k+1}$ единиц, и отвергает его, если среди её аргументов больше $2^{l_2+q-m+k+1.01}$ единиц. Поскольку все уровни имеют константную глубину и размер $2^{\mathrm{poly}(n)}$, то и вся схема обладает этим свойством, что и требовалось получить. Эскиз схемы показан на рис. 6.3.

Докажем корректность построенной схемы. Проведём рассуждение для одного блока, построенного для конкретной тройки $(S_1, S_2, \mathbf{Q})$. Пусть он получил на вход код $(k, q, 1, \mathcal{R}, \mathcal{S})$-равномерно пёстрой таблицы. Значит, общее количество ячеек в насыщенных строках в $S_1 \times S_2$ не больше $2^{l_1+q-m+k+1}$. Если строка y_i насыщена, то в ней больше $2^{l_1+q-m+1}$ ячеек покрашены в цвета из Q_i. Для ненасыщенных строк пороговая схема на третьем уровне точно выдаст 0, а для насыщенных — 1 или 0 (последнее возможно, если помеченных ячеек больше $2^{l_1+q-m+1}$, но меньше $2^{l_1+q-m+1.01}$). Далее, единицы на четвёртом уровне соответствует помеченным ячейкам, про строки которых предыдущая схема выдала 1. Все эти ячейки помечены и находятся в насыщенных строках, значит их не больше $2^{l_1+q-m+k+1}$. Значит, последняя пороговая схема выдаст 1, что и требовалось.

Пусть, наоборот, схема выдала 1. Значит, единиц на четвёртом уровне не больше $2^{l_1+q-m+k+1.01}$. Каждая единица соответствует помеченной ячейке, расположенной в строке, про которую схема на третьем уровне выдала 1. Все насыщенные (для $b = 1.01$) строки в это список попали, т.к. для них пороговая схема на третьем уровне точно выдаёт 1. Значит, помеченных ячеек в насыщенных строках не больше $2^{l_1+q-m+k+1.01}$. Поскольку это верно для всех троек $(S_1, S_2, \mathbf{Q})$, таблица является $(k, q, 1, \mathcal{R}, \mathcal{S})$-равномерно пёстрой по стркоам. Аналогично доказывается $(k, q, 1, \mathcal{R}, \mathcal{S})$-равномерная пестрота по столбцам.

Таким образом, заявленные схемы $\mathcal{BT}$ и $\mathcal{RBT}$ построены. □

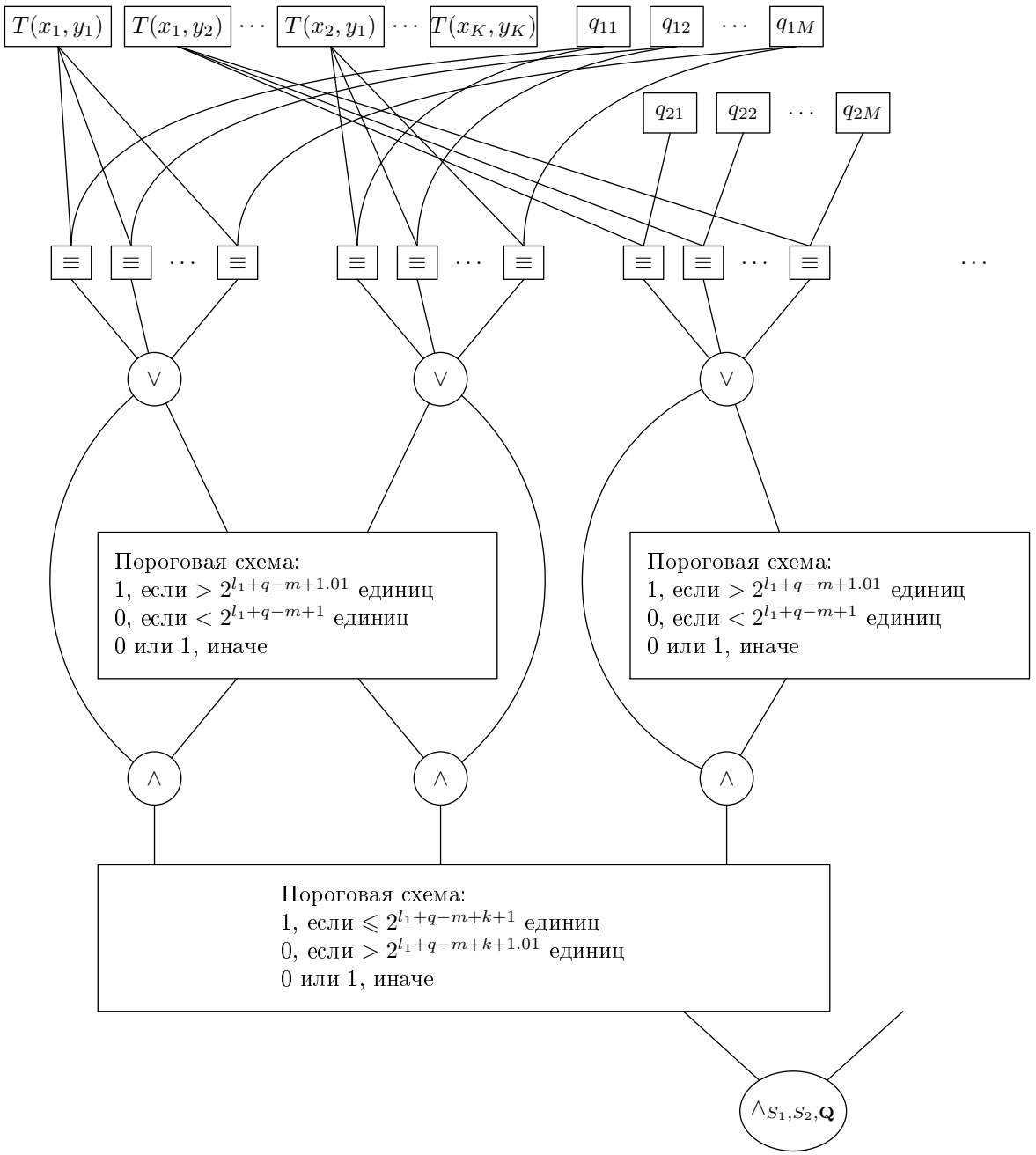


Рис. 6.3: Эскиз схемы, проверяющей равномерную пестроту таблицы.

Применив лемму 2.37, получаем следующее

Следствие 6.15. *Для заданных ранее значений параметров и любых релевантных систем $\mathcal{R}$ и $\mathcal{S}$ и множества Q среди значений генератора Нисана–Вигдерсона найдутся коды $(k, q, 1.01, Q, \mathcal{S})$-пёстрой и $(k, q, 1.01, \mathcal{R}, \mathcal{S})$-равномерно пёстрой таблиц.*

6.5 Завершение доказательства основной теоремы

В этом разделе мы опишем, как найти таблицы, описанные в предыдущем разделе (точнее, аргументы генератора, которые нужны для их порождения), и покажем, что найденные таблицы действительно являются колмогоровскими экстракторами. Здесь нам потребуется определить конкретные системы $\mathcal{R}$ и $\mathcal{S}$ и множество Q, для которых мы будем проводить рассуждения.

Определение 6.16. Пусть фиксированы числа $k < n$ и $q < m$. Здесь и далее будем полагать, что:

- $\mathcal{S}$ есть система всех пар $(\{z \in \{0,1\}^n \mid \mathrm{C}^s(z) < l\}, l)$ для $s = s(n)$ и всех $l \in [k, n]$;
- $Q = \{w \in \{0,1\}^m \mid \mathrm{C}^s(w) < q\}$ для $s = s(n)$;
- $\mathcal{R}$ есть система всех кортежей

$$\left(\left\{w \in \{0,1\}^m \mid \mathrm{C}^s(w|v_1) < q\right\}, \ldots, \left\{w \in \{0,1\}^m \mid \mathrm{C}^s(w|v_L) < q\right\}, \underbrace{\varnothing, \ldots, \varnothing}_{2^l - L \text{ штук}}\right) \tag{6.2}$$

для $s = s(n)$ и всех $\big(\{v_1, \ldots, v_L\}, l\big) \in \mathcal{S}$, где порядок $v_1, \ldots, v_L$ считается фиксированным при выборе соответствующего множества.

Утверждение 6.17. *Заданные в определении 6.16 системы $\mathcal{S}$ и $\mathcal{R}$ и множество Q удовлетворяют требованиям определений 6.7 и 6.10. Кроме того, все эти системы перечислимы на памяти $O(s+n)$, т.е. существуют алгоритмы, работающие на памяти $O(s+n)$ и делающие следующее:*

- *Для $\mathcal{S}$: во-первых, по индексу i алгоритм возвращает вторую компоненту i-ой пары из $\mathcal{S}$; во-вторых, по индексам i и j алгоритм возвращает j-ый элемент первой компоненты i-ой пары из $\mathcal{S}$.*
- *Для Q: по индексу j алгоритм возвращает j-ый элемент Q.*
- *Для $\mathcal{R}$: по индексам i, j и k алгоритм возвращает k-ый элемент j-ой компоненты i-го кортежа из $\mathcal{R}$.*

Во всех случаях алгоритм возвращает символ ошибки, если хотя бы один из индексов превысил размер соответствующего множества.

Доказательство. Действительно, условия на размеры выполнены:

- Система $\mathcal{S}$ содержит по одной паре для каждого $l \in [k, n]$, т.е. всего не больше n пар. Каждая пара имеет вид (S, l), где множество S содержит меньше 2^l элементов, поскольку все его элементы имеют сложность меньше l.

- Множество Q содержит меньше 2^q элементов, поскольку все его элементы имеют сложность меньше q.

- Система $\mathcal{R}$ содержит по одному кортежу для каждой пары $(S, l) \in \mathcal{S}$, поэтому её размер также не больше n. Длина соответствующего кортежа равна 2^l по построению, а каждое множество из кортежа содержит меньше 2^q элементов, поскольку все его элементы имеют условную сложность меньше q.

Утверждение о перечислимости следует из того, что множество всех слов, у которых (условная) сложность с ограничением на память s меньше фиксированного числа, перечислимо на памяти $O(s + n)$. Действительно, сложность C^s можно вычислить на памяти $O(s + n)$, поэтому для перечисления всех слов заданной длины, имеющих сложность не больше фиксированной, достаточно перечислять все слова заданной длины, считать сложность у каждого из них и оставлять только нужные. Подробнее:

- Для $\mathcal{S}$: поскольку l пробегает все индексы от k до n, то вторую компоненту пары номер i можно положить равной $k + i - 1$. Для фиксированного i нужно перечислять все слова длины n, имеющие сложность меньше $k + i - 1$.

- Для Q: нужно перечислять все слова длины m, имеющие сложность меньше q.

- Для $\mathcal{R}$: по i и j находится слово v_j из соответствующего множества, затем перечисляются все слова длины m, имеющие условную сложность относительно v_j меньше q.

□

Теперь докажем, что аргумент генератора Нисана–Вигдерсона, порождающий нужную таблицу, можно найти на небольшой памяти:

Теорема 6.18. *Пусть фиксированы числа $k < n$ и $q < m$, а системы $\mathcal{S}$ и $\mathcal{R}$ и множество Q заданы в соответствии с определением 6.16. Пусть $G_n\colon \{0,1\}^{r(n)} \to \{0,1\}^{m2^{2n}}$ есть генератор, существующий в соответствии со следствием 2.36. Тогда найдётся алгоритм, получающий на вход параметры n, m, k, q, работающий на памяти $O(s) + \mathrm{poly}(n)$ и возвращающий слова u_{BT} и u_{RBT} длины $r(n)$, для которых:*

- *$G_n(u_{BT})$ есть код некоторой $(k, q, 1.01, Q, \mathcal{S})$-пёстрой таблицы;*
- *$G_n(u_{RBT})$ есть код некоторой $(k, q, 1.01, \mathcal{R}, \mathcal{S})$-равномерно пёстрой таблицы.*

Доказательство. Поскольку для вычислений с ограничением на память задача поиска эквивалентна задаче распознавания, то мы опишем, как по слову $u \in \{0,1\}^{r(n)}$ распознать, является ли $G_n(u)$ кодом некоторой (равномерно) пёстрой таблицы. Ключевая особенность генератора Нисана–Вигдерсона, которая будет использована при доказательстве — это возможность вычислить любой заданный бит выхода, используя память $\mathrm{poly}(r(n)) = \mathrm{poly}(n)$.

Вначале проведём рассуждение для $(k, q, 1.01, Q, \mathcal{S})$-пёстрых таблиц. Пусть задано слово $u \in \{0,1\}^{r(n)}$. Обозначим через T таблицу, закодированную словом $G_n(u)$. Требуется проверить, что для любых пар (S_1, l_1) и (S_2, l_2) из $\mathcal{S}$ количество ячеек в $S_1 \times S_2$, покрашенных в цвет из множества Q, меньше $2^{l_1+l_2+q-m+1.01}$. Поскольку при нашем выборе $\mathcal{S}$ множества S_1 и S_2 однозначно определяются числами l_1 и l_2, распознающий алгоритм будет перебирать все пары (l_1, l_2) и проверять выполнения условия для каждой пары. Для этого алгоритм заведёт счётчик c и будет перечислять множество $S_1 \times S_2$ (это можно сделать, поскольку S_1 и S_2 перечислимы), для каждой полученной пары $(x, y) \in S_1 \times S_2$ вычислять $T(x, y)$ и считать сложность $\mathrm{C}^s(T(x, y))$. Если сложность будет меньше q (это означает, что ячейка (x, y) покрашена в цвет из Q), то он увеличит счётчик c на единицу. Если в конце перебора счётчик не превысил $2^{l_1+l_2+q-m+1.01}$, то пара (S_1, S_2) прошла тест, и алгоритм переходит к следующей паре. Иначе цикл заканчивается и таблица T и породившее её слово u отвергаются. Если же все пары (S_1, S_2) прошли тест, то T и u принимаются. Алгоритм 6.1 показывает псевдокод.

Корректность алгоритма следует из конструкции. Проверим, что ограничение на память соблюдается. Действительно, между шагами (внутрен-

Вход: Числа n, m, k и q
Выход: Слово u, такое что $G_n(u)$ кодирует $(k, q, 1.01, Q, \mathcal{S})$-пёструю таблицу
$r := O(n^{2d+6})$; /* Точное значение как в следствии 2.36 */
для каждого $u \in \{0,1\}^r$ **выполнить**
  для каждого $(l_1, l_2) \in [k, n]^2$ **выполнить**
    count $:= 0$;
    для каждого $(x, y) \in (\{0,1\}^n)^2$ **выполнить**
      если $\mathrm{C}^s(x) < l_1$, $\mathrm{C}^s(y) < l_2$ u $\mathrm{C}^s(T(x,y)) < q$ **то** count $++$;
      /* Для вычисления T берутся нужные биты $G_n(u)$ */
    конец цикла
    если count $> 2^{l_1+l_2+q-m+1.01}$ **то продолжить цикл** по u;
  конец цикла
  возвратить u ; /* Выполняется, только если для всех l_1 и l_2 выполнено count $\leqslant 2^{l_1+l_2+q-m+1.01}$ */
конец цикла

Алгоритм 6.1: Поиск аргумента генератора Нисана–Вигдерсона для получения пёстрой таблицы.

него) цикла требуется хранить только значение счётчика и текущие значения l_1, l_2, x, y, т.е. $\mathrm{poly}(n)$ битов. Каждую следующую пару из $S_1 \times S_2$ можно найти на памяти $O(s+n)$ в силу утверждения 6.17. Далее, $T(x,y)$ вычисляется на памяти $\mathrm{poly}(n)$ в силу свойств генератора Нисана–Вигдерсона: для вычисления $T(x,y)$ нужно найти m битов $G_n(u)$, каждый из которых можно найти на памяти $\mathrm{poly}(n)$. Наконец, сложность $\mathrm{C}^s(T(x,y))$ также вычисляется на памяти $O(s+n)$. Таким образом, все вычисления требуют памяти $O(s) + \mathrm{poly}(n)$, что и требовалось.

Теперь проведём рассуждение для $(k, q, 1.01, \mathcal{R}, \mathcal{S})$- равномерно пёстрых таблиц. Мы проведём рассуждение только для равномерно по столбцам пёстрых таблиц, поскольку рассуждение для равномерно по строкам пёстрых таблиц полностью аналогично. Как и прежде, зададимся словом $u \in \{0,1\}^{r(n)}$ и обозначим через T таблицу, закодированную словом $G_n(u)$. Как и прежде, достаточно описать процедуру для конкретных пар (S_1, l_1) и (S_2, l_2) и кортежа $\mathbf{Q}$, заданного S_1. Требуется посчитать количество отмеченных ячеек в насыщенных столбцах и сравнить это число с $2^{l_2+q-m+k+1.01}$. Для этого алгоритм заводит счётчик c_1, последовательно перебирает все столбцы (т.е. элементы $x \in S_1$), считает отмеченные ячейки в каждом столбце и, если это число превысило $2^{l_2+q-m+1.01}$, добавляет это число к счётчику c_1. Подсчёт отмеченных ячеек в одном столбце проводится так: заводится ещё один счётчик c_2, перебираются все $y \in S_2$, для каждого вы-

Вход: Числа n, m, k и q
Выход: Слово u, такое что $G_n(u)$ кодирует $(k, q, 1.01, \mathcal{R}, \mathcal{S})$-пёструю таблицу

```
1  r := O(n^{2d+6});                          /* Точное значение как в следствии 2.36 */
2  для каждого u ∈ {0,1}^r выполнить
3      для каждого (l_1, l_2) ∈ [k, n]^2 выполнить
4          count_1 := 0;
5          для каждого x ∈ {0,1}^n выполнить
6              если C^s(x) < l_1 то
7                  count_2 := 0;
8                  для каждого y ∈ {0,1}^n выполнить
9                      если C^s(y) < l_2 и C^s(T(x,y)|x) < q то count_2 ++;
                       /* Для вычисления T берутся нужные биты G_n(u)          */
10                 конец цикла
11                 если count_2 > 2^{l_2+q-m+1.01} то count_1 += count_2;
12             конец условия
13         конец цикла
14         если count_1 > 2^{l_2+q-m+k+1.01} то продолжить цикл по u;
15     конец цикла
16     возвратить u ;   /* Выполняется, только если для всех l_1 и l_2 выполнено
           count ⩽ 2^{l_2+q-m+k+1.01} */
17 конец цикла
```

Алгоритм 6.2: Поиск аргумента генератора Нисана–Вигдерсона для получения равномерно пёстрой таблицы.

числяются значение $T(x, y)$ и сложность $\mathrm{C}^s(T(x, y)|x)$. Если эта сложность меньше q, то счётчик c_2 увеличивается на 1. Алгоритм 6.2 показывает псевдокод.

Корректность этого алгоритма вновь следует из конструкции. Ограничение на память также соблюдается: между шагами (внутреннего) цикла требуется хранить только значения счётчиков и текущие значения l_1, l_2, x, y, т.е. $\mathrm{poly}(n)$ битов. (Выбор l_1 и l_2 однозначно определяет не только S_1 и S_2, но и $\mathbf{Q}$). Следующие по очереди x и y можно найти на памяти $O(s+n)$ в силу утверждения 6.17; $T(x, y)$ вычисляется на памяти $\mathrm{poly}(n)$ в силу свойств генератора Нисана–Вигдерсона; сложность $\mathrm{C}^s(T(x, y))$ также вычисляется на памяти $O(s + n)$. Таким образом, все вычисления требуют памяти $O(s) + \mathrm{poly}(n)$, что и требовалось. □

Осталось доказать, что найденные в теореме 6.18 таблицы $G_n(u_{BT})$ и $G_n(u_{RBT})$ являются колмогоровскими экстракторами (последняя — в сильном смысле). Это завершит доказательство теоремы 6.2.

Теорема 6.19. *Существуют полином $p(n)$ и константа c, для которых верно следующее. Для $\delta = m - q - c\log n$ и любой конструируемой по памяти функции $s(n) > p(n)$ построенная в теореме 6.18 $(k, q, 1.01, Q, \mathcal{S})$-пёстрая таблица $G_n(u_{BT})$ является (k, δ)-колмогоровским экстрактором для ограничения на память $s = s(n)$, а построенная в той же теореме $(k, q, 1.01, \mathcal{R}, \mathcal{S})$- равномерно пёстрая таблица $G_n(u_{RBT})$ является (k, δ)-колмогоровским экстрактором в сильном смысле для того же ограничения на память.*

Доказательство. Во-первых, заметим, что эти таблицы вычислимы на памяти $O(s(n))$: при выборе достаточно большого $p(n)$ этой памяти достаточно как для поиска слов u_{BT} и u_{RBT} (в силу теоремы 6.18), так и для вычисления цветов ячеек соответствующих таблиц, т.е. отдельных битов $G_n(u_{BT})$ и $G_n(u_{RBT})$ (в силу свойства генератора Нисана–Вигдерсона).

Во-вторых, докажем, что для $T = G_n(u_{BT})$ выполнено свойство колмогоровского экстрактора. Действительно, пусть даны слова x и y длины n, для которых $\mathrm{C}^s(x) = l_1 > k$, $\mathrm{C}^s(y) = l_2 > k$ и $\mathrm{C}^{\mu s}(x, y) > l_1 + l_2 - \delta$. (Константа μ не зависит от x и y и будет выбрана позднее). Предположим, что для этих слов свойство колмогоровского экстрактора не выполнено, т.е. $\mathrm{C}^s(T(x, y)) \leqslant m - \delta - \Omega(\log n)$. Можно считать, что константа, скрытая в $\Omega(\log n)$, больше c, поэтому $\mathrm{C}^s(T(x, y)) < q$. В этом случае $T(x, y) \in Q$ по определению Q. Далее, введём обозначения $S_1 = \{z \mid \mathrm{C}^s(z) < l_1\}$, $S_2 = \{w \mid \mathrm{C}^s(w) < l_2\}$. По построению пары (S_1, l_1) и (S_2, l_2) лежат в $\mathcal{S}$, поэтому в силу $(k, q, 1.01, Q, \mathcal{S})$-пестроты T в прямоугольнике $S_1 \times S_2$ не больше $2^{l_1+l_2+q-m+1.01}$ ячеек, покрашенных в цвет из Q. При этом в силу нашего предположения ячейка (x, y) относится к числу этих ячеек. Это значит, что ячейку (x, y) можно описать, задав числа n, l_1, l_2, q и её порядковый номер среди всех ячеек, обладающих данным свойством. Действительно, по n можно найти u_{BT}, затем по l_1 и l_2 можно перечислять $S_1 \times S_2$ и, наконец, по q проверять, верно ли что $\mathrm{C}^s(T(x, y)) < q$, и перечислять те пары, для которых это верно. Порядковый номер в перечислении задаст нужную пару. Общее число битов этого описания не превосходит $4\log n + (l_1 + l_2 + q - m + 1.01) + O(1) < l_1 + l_2 - \delta + (5 - c)\log n$, при этом используемая память равна $O(s)$. Таким образом, при наших предположениях

$$\mathrm{C}^{O(s)}(x, y) < l_1 + l_2 - \delta + (5 - c)\log n = \mathrm{C}^s(x) + \mathrm{C}^s(y) - \delta + (5 - c)\log n, \quad (6.3)$$

что противоречит условию $\mathrm{C}^{\mu s}(x,y) > \mathrm{C}^s(x) + \mathrm{C}^s(y) - \delta$ при $c > 5$ и достаточно большом μ.

В-третьих, докажем, что для $T = G_n(u_{RBT})$ выполнены свойства усиленного колмогоровского экстрактора. Приведём рассуждение только для одного из условий, второе получится симметрично. Пусть даны слова x и y длины n, для которых $\mathrm{C}^s(x) = l_1 > k$, $\mathrm{C}^s(y) = l_2 > k$ и $\mathrm{C}^{\mu s}(x,y) > l_1 + l_2 - \delta$. (Константа μ не зависит от x и y и будет выбрана позднее). Предположим, что для этих слов свойство усиленного колмогоровского экстрактора не выполнено, т.е. $\mathrm{C}^s(T(x,y)|x) \leqslant m - \delta - \Omega(\log n)$. Можно считать, что константа, скрытая в $\Omega(\log n)$, больше c, поэтому $\mathrm{C}^s(T(x,y)|x) < q$. Определим множества S_1 и S_2 так же, как в рассуждении для обычных экстракторов, вновь получим, что (S_1, l_1) и (S_2, l_2) лежат в $\mathcal{S}$. Определим кортеж $\mathbf{Q}$ через множество $S_1 = \{v_1, \ldots, v_L\}$ согласно (6.2). Тогда, очевидно, $\mathbf{Q} \in \mathcal{R}$. Далее, рассмотрим два случая: либо существует больше $2^{l_2+q-m+1.01}$ слов $z \in S_2$, удовлетворяющих условию $\mathrm{C}^s(T(x,z)|x) < q$, либо существует не больше $2^{l_2+q-m+1.01}$ таких слов.

В первом случае столбец $\{x\} \times S_2$ будет насыщенным, поскольку содержит больше $2^{l_2+q-m+1.01}$ помеченных (как в определении 6.11) ячеек, при этом ячейка (x,y) помечена. В силу $(k, q, 1.01, \mathcal{R}, \mathcal{S})$-равномерной пестроты таблицы T общее количество помеченных ячеек в насыщенных столбцах не больше $2^{l_1+l_2+q-m+k+1.01}$, что меньше $2^{l_1+l_2-\delta-c\log n+1.01}$. Таким образом, ячейка (x,y) может быть задана числами n, l_1, l_2, q и своим порядковым номером среди отмеченных ячеек в насыщенных столбцах, причём для поиска (x,y) по этим данным требуется память $O(s)$. Таким образом, $\mathrm{C}^{O(s)}(x,y) < 4\log n + l_1 + l_2 - \delta - c\log n + 1.01 + O(1) < l_1 + l_2 - \delta + (5-c)\log n$, что противоречит условию $\mathrm{C}^{\mu s}(x,y) > \mathrm{C}^s(x) + \mathrm{C}^s(y) - \delta$ при $c > 5$ и достаточно большом μ.

Во втором случае пара (x,y) может быть получена на памяти $O(s)$, если известны числа n, l_1, l_2, q (не больше $4\log n$ битов), кратчайшее описание x (l_1 битов) и номер y среди помеченных ячеек в столбце $\{x\} \times S_2$ (не больше $l_2 + q - m + 1.01 = l_2 - \delta - c\log n + 1.01$ битов). В совокупности эти данные занимают $4\log n + l_1 + l_2 - \delta - c\log n + 1.01$ битов, т.е. мы получаем то же противоречие, что и в первом случае.

Таким образом, утверждение полностью доказано. □

Для завершения доказательства теоремы 6.2 проведём оценку парамет-

ров. По теореме 6.5 существуют пёстрые таблицы для $k \geqslant m/2+O(\log n)$ и $q \geqslant m/2+O(\log n)$. Согласно утверждению 6.9 существуют и $(k, q, 1, Q, \mathcal{S})$-пёстрые таблицы для этих параметров. Значит, по теореме 6.18 найдётся алгоритм, находящий u_{BT}, для которого $G_n(u_{BT})$ есть код некоторой $(k, q, 1.01, Q, \mathcal{S})$-пёстрой таблицы, а по теореме 6.19 эта таблица является $(k, m-q-c\log n)$-колмогоровским экстрактором. Путём несложных преобразований получаем, что годятся любые $m \leqslant 2k - O(\log n)$ и $\delta = m - q - c\log n \leqslant m - m/2 - O(\log n) - c\log n \leqslant k - O(\log n)$, что соответствует утверждению теоремы 6.2.

Аналогично по теореме 6.5 существуют равномерно пёстрые таблицы для $k \geqslant m + O(\log n)$ и любого q. Согласно утверждению 6.13 существуют и $(k, q, 1.01, \mathcal{R}, \mathcal{S})$-равномерно пёстрые таблицы для этих параметров. Значит, по теореме 6.18 найдётся алгоритм, находящий u_{RBT}, для которого $G_n(u_{RBT})$ есть код некоторой $(k, q, 1.01, \mathcal{R}, \mathcal{S})$-равномерно пёстрой таблицы, а по теореме 6.19 эта таблица является $(k, m-q-c\log n)$-колмогоровским экстрактором в сильном смысле. Путём несложных преобразований получаем, что годятся любые $m \leqslant k - O(\log n)$ и $\delta = m-q-c\log n \leqslant m-c\log n \leqslant k-O(\log n)$, что соответствует утверждению теоремы 6.2.

Таким образом, основной результат этой главы — теорема 6.2 — полностью доказан.

6.6 Теорема для малых ограничений на память

Наши аналоги теоремы Мучника были сформулированы для произвольного ограничения на память s, а теорема 6.2 только для не менее чем полиномиального, причём довольно большого. Это связано с двумя вещами. Во-первых, нахождение аргумента генератора Нисана–Вигдерсона требует полиномиальной памяти, т.е. для вычисления построенного колмогоровского экстрактора требуется полиномиальная память. Во-вторых, противоречивые оценки на сложность ячейки (x, y) получаются только при достаточно большом полиномиальном ограничении, которое требуется для короткого описания. Тем не менее, требование полиномиальности s можно ослабить и доказать следующее усиление теоремы 6.2. Чтобы не менять определение колмогоровского экстрактора с ограничением на память, напишем все утверждения явно.

Теорема 6.20. *Пусть заданы числа n, s, $k \in (1, n)$ и $\delta \in (1, k - O(\log n))$. Тогда для некоторых полиномов $p_1(n)$ и $p_2(n)$ и константы μ существуют функции:*

- $T\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, *где* $m = 2k - O(\log n)$, *вычислимая на памяти* $p_1(n)$, *для которой выполнено следующее свойство. Для любых слов* x *и* y *длины* n, *для которых выполнено* $\mathrm{C}^s(x) > k$, $\mathrm{C}^s(y) > k$ *и* $\mathrm{C}^{\mu s + p_2(n)}(x, y) > \mathrm{C}^s(x) + \mathrm{C}^s(y) - \delta$, *верно* $\mathrm{C}^s(T(x, y)) > m - \delta - O(\log n)$.
- $V\colon \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$, *где* $m = k - O(\log n)$, *вычислимая на памяти* $p_1(n)$, *для которой выполнено следующее свойство. Для любых слов* x *и* y *длины* n, *для которых выполнено* $\mathrm{C}^s(x) > k$, $\mathrm{C}^s(y) > k$ *и* $\mathrm{C}^{\mu s + p_2(n)}(x, y) > \mathrm{C}^s(x) + \mathrm{C}^s(y) - \delta$, *верно* $\mathrm{C}^s(V(x, y)|x) > m - \delta - O(\log n)$ *и* $\mathrm{C}^s(V(x, y)|y) > m - \delta - O(\log n)$.

Доказательство. Доказательство этой теоремы повторяет основные шаги доказательства теоремы 6.2 и использует те же леммы. Поэтому мы не будем его приводить целиком, а остановимся на двух основных отличиях. Во-первых, поскольку s больше не обязано быть полиномиальным, функции T и V могут не быть вычислимы на памяти $O(s)$. Поэтому мы вводим полиномиальное ограничение $p_1(n)$ в явном виде. Во-вторых, при доказательстве аналога теоремы 6.19 в неравенстве (6.3) больше нельзя использовать ограничение на память $O(s)$, вместо этого должно быть $O(s)+\mathrm{poly}(n)$. Поэтому помимо константы μ мы вводим в формулировку полином $p_2(n)$. □

Также можно заметить, что любая явная комбинаторная конструкция (равномерно) пёстрой таблицы приводит к доказательству теоремы 6.2 (и теоремы 6.20). Однако, насколько известно автору, таких конструкций пока не создано.

Заключение

В статье [6] Колмогоров говорил о трёх подходах к понятию «количество информации»: комбинаторном, вероятностном и алгоритмическом. Каждый из этих подходов представляет интерес и довольно подробно изучен, но особенно интересны соотношения между разными подходами. В настоящей работе мы установили новые связи между комбинаторными и алгоритмическими утверждениями и разработали новый способ распространения утверждений об обычной колмогоровской сложности на сложность с ограничением на ресурсы. Однако множество вопросов остаются нерешёнными. Прежде всего: каковы пределы нашего подхода? Какие утверждения о колмогоровской сложности можно переложить для сложности с ограничением на ресурсы через посредничество комбинаторных утверждений? Насколько универсален метод «наивной дерандомизации»? Можно ли доказать какую-либо общую теорему на этот счёт? Эти вопросы являются предметом дальнейшего изучения.

Представляет интерес вопрос о том, можно ли построить колмогоровский аналог для экстракторов с одним источником подобно тому, как колмогоровские экстракторы похожи на экстракторы с двумя источниками. Возможно, теорема Мучника может пролить свет на этот вопрос.

Ещё одно важное направление развития — построение явных конструкций использованных нами комбинаторных объектов. Например, построение явных экстракторов с оптимальными параметрами стало бы прорывом сразу во многих областях. Для наших целей интересно было бы построить экстрактор с оптимальными параметрами, вычислимый на полиномиальной памяти: тогда бы для соответствующего аналога теоремы Мучника не нужно было бы наивной дерандомизации. Возможно, наоборот, методом наивной дерандомизации можно построить такой экстрактор. Можно ли при помощи каких-либо явных экстракторов доказать теорему Мучника для обычной или CBP-сложности с ограничением на память вместо CAM-

сложности? Возможно, это как-то связано со стандартными теоретико-сложностными предположениеми? Также отличным результатом стало бы построение полиномиально вычислимых пёстрых таблиц: возможно, это позволило бы доказать существование колмогоровских экстракторов для сложности с ограничением на время (хотя бы для CAM-сложности). Также открытым вопросом остаётся справедливость гипотезы 5.8 о теореме Мучника с двумя условиями для CAM-сложности.

Можно сказать, что теория колмогоровской сложности с ограничением на ресурсы всё ещё находится в стадии накопления фактов. Довести количество известных результатов до критической массы и уложить их в стройную теорию остаётся задачей будущих исследований.

Литература

1. **Алон, Н.** Вероятностный метод в комбинаторике / Н. Алон, Дж. Спенсер. — М.: БИНОМ, 2011. — 320 с.

2. **Верещагин, Н.К.** Колмогоровская сложность и алгоритмическая случайность / Н.К. Верещагин, В.А. Успенский, А. Шень. — М.: МЦНМО, 2013. — 576 с.

3. **Звонкин, А.К.** Сложность конечных объектов и обоснование понятий информации и случайности с помощью теории алгоритмов / А.К. Звонкин, Л.А. Левин // Успехи математических наук. — 1970. — Т. 25. №3. — С. 85–127.

4. **Ирхин, И.А.** Экстракторы и почти экстракторы для двух источников: выпускная квалификационная работа / И.А. Ирхин. — М.: МФТИ, 2014. — 33 с.

5. **Ицыксон, Д.М.** Вероятностные методы в вычислениях: краткий конспект лекций / Д.М. Ицыксон. — `http://logic.pdmi.ras.ru/csclub/sites/default/files/random.pdf` — 2012.

6. **Колмогоров, А.Н.** Три подхода к определению понятия «Количество информации» / А.Н. Колмогоров // Проблемы передачи информации. — 1965. — Т.1. №1. — С. 3–11.

7. **Международный математический турнир городов.** Условия двадцать седьмого Турнира Городов, 2005–2006. / `http://www.mccme.ru/turgor/27/` — 2006.

8. **Ajtai, M.** Approximate counting with uniform constant-depth circuits / M. Ajtai // Advances in computational complexity theory — American Mathematical Society, 1993. — P. 1–20.

9. **Alon, N.** Simple constructions of almost k-wise independent random variables / N. Alon, O. Goldreich, J. Håstad, R. Peralta // Random Structures and Algorithms. — 1992. — Vol. 3, no. 3. — P. 289–303.

10. **Arora, S.** Computational Complexity: A Modern Approach / S. Arora, B. Barak — Cambridge University Press, 2007. — 579 p.

11. **Babai, L.** Trading Group Theory for Randomness / L. Babai // Proceedings of the 17th annual ACM symposium on Theory of computing. — 1985. — P. 421–429.

12. **Buhrman, H.** Resource bounded Kolmogorov complexity revisited / H. Buhrman, L. Fortnow, S. Laplante // SIAM Journal on Computing. — 2002. — Vol. 31, no. 3. — P. 887–905.

13. **Buhrman, H.** Language Compression and Pseudorandom Generators / H. Buhrman, T. Lee, D. van Melkebeek // Computational Complexity. — 2005. — Vol. 14. — P. 247–274.

14. **Chaitin, G.J.** On the length of programs for computing binary sequences / G.J. Chaitin // Journal of the ACM — 1966. — Vol. 13, no. 4. — P. 547–569.

15. **Chaitin, G.J.** On the length of programs for computing binary sequences: statistical considerations / G.J. Chaitin // Journal of the ACM. — 1969. — Vol. 16, no. 1. — P. 145–159.

16. **Chor, B.** Unbiased Bits From Sources of Weak Randomness and Probabilistic Communication / B. Chor, O. Goldreich // SIAM Journal on Computing. —1988. — Vol. 17, no. 2. — P. 230–261.

17. **Dvir, Z.** Kakeya Sets, New Mergers, and Old Extractors / Z. Dvir, A. Wigderson // SIAM Journal of Computing. — 2011. — Vol. 40, no. 3. — P. 778–792.

18. **Fortnow, L.** Extracting Kolmogorov Complexity with Applications to Dimension Zero-One Laws / L. Fortnow, J. Hitchcock, A. Pavan, N.V. Vinodchandran, F. Wang // Information and Computation. — 2011. — Vol. 209, no. 4. — P. 627–636.

19. **Furst, M.** Parity, Circuits, and the Polynomial-Time Hierarchy / M. Furst, J.B. Saxe, M. Sipser // Mathematical systems theory. — 1984. — Vol. 17, no. 1. — P. 13–27.

20. **Goldreich, O.** Three XOR-Lemmas — an Exposition / O. Goldreich // Studies in Complexity and Cryptography. — 2011. — Springer LNCS Vol. 6650. — P. 248–272.

21. **Guruswami, V.** Unbalanced Expanders and Randomness Extractors from Parvaresh-Vardy Codes / V. Guruswami, C. Umans, S. Vadhan // Journal of the ACM. — 2009. — Vol. 56, no. 4. — P. 20:1–20:34.

22. **Hitchcock, J.** Kolmogorov Complexity in Randomness Extraction / J.M. Hitchcock, A. Pavan, N.V. Vinodchandran // ACM Transactions on Computation Theory. — 2011. — Vol. 3, no. 1. — P. 1:1–1:??.

23. **Hoeffding, W.** Probability Inequalities for Sums of Bounded Random Variables / W. Hoeffding // Journal of the American Statistical Association. — 1963. — Vol. 58, no. 301. — P. 13–30.

24. **Impagliazzo, R.** Pseudo-Random Generation from One-Way Functions / R. Impagliazzo, L.A. Levin, M. Luby // Proceedings of the 21st Annual ACM Symposium on the Theory of Computing. — 1989. — P. 12–24.

25. **Impagliazzo, R.** Hardness as randomness: A survey of universal derandomization / R. Impagliazzo // Proceedings of the ICM. — 2002. — Vol. 3. — P. 659–672.

26. **Kabanets, V.** Derandomization: A Brief Overview / V. Kabanets // Current Trends in Theoretical Computer Science: The Challenge of the New Century, Vol 1: Algorithms and Complexity. — World Scientific Publishing Company, 2004. — P. 165–187.

27. **Lee, T.** Resource-Bounded Symmetry of Information Revisited / T. Lee, A. Romashchenko // Theoretical Computer Science. — 2005. — Vol. 345, no. 2–3. — P. 386–405.

28. **Li, M.** An Introduction to Kolmogorov Complexity and its Applications / M. Li, P.M.B. Vitányi. — 3rd Edition. — Springer, 2008. — XXIV, 792 p.

29. **Lu, C.-J.** Extractors: Optimal up to Constant Factors / C.-J. Lu, O. Reingold, S. Vadhan, A. Wigderson // Proceedings of the 35th Annual ACM Symposium on the Theory of Computing. — 2003. — P. 602–611.

30. **Muchnik, An.A.** Conditional Complexity and Codes / An.A. Muchnik // Theoretical Computer Science. — 2002. — Vol. 271, no. 1–2. — P. 97–109.

31. **Naor, J.** Small-Bias Probability Spaces: Efficient Constructions and Applications / J. Naor, M. Naor // SIAM Journal of Computing. — 1993. — Vol. 22, no. 4. — P. 838–856.

32. **Nisan, N.** Pseudorandom Bits for Constant Depth Circuits / N. Nisan // Combinatorica. — 1991. — Vol. 11, no. 1. P. 63–70.

33. **Nisan, N.** Extracting Randomness: A Survey and New Constructions / N. Nisan, A. Ta-Shma // Journal of Computer and System Sciences. — 1999. — Vol. 58, no. 1. — P. 148–173.

34. **Nisan, N.** Hardness vs. Randomness. / N. Nisan, A. Wigderson // Journal of Computer and System Sciences. — 1994. — Vol. 49. — P. 149–167.

35. **Nisan, N.** Randomness is Linear in Space / N. Nisan, D. Zuckerman // Journal of Computer and System Sciences. — 1996. — Vol. 52, no. 1. — P. 43–52.

36. **Radhakrishnan, J.** Bounds for Dispersers, Extractors, and Depth-Two Superconcentrators / J. Radhakrishnan, A. Ta-Shma // SIAM Journal on Discrete Mathematics. — 2000. — Vol. 13, no. 1. — P. 2–24.

37. **Raz, R.** Extracting All the Randomness and Reducing the Error in Trevisan's Extractor / R. Raz, O. Reingold, S. Vadhan // Proceedings of the 30th Annual ACM Symposium on the Theory of Computing. — 1999. — P. 149–158.

38. **Reingold, O.** Extracting Randomness via Repeated Condensing / O. Reingold, R. Shaltiel, A. Wigderson // SIAM Journal on Computing. — 2006. — Vol. 35, no. 5. — P. 1185–1209.

39. **Romashchenko, A.E.** Pseudo-Random Graphs and Bit Probe Schemes with One-Sided Error / A.E. Romashchenko // Theory of Computing Systems. — 2014. — Vol. 55, no. 2. — P. 313–329.

40. **Shaltiel, R.** Recent Developments in Explicit Constructions of Extractors / R. Shaltiel // Current and Trends in Theoretical Computer Science: The Challenge of the New Century. Volume 1: Algorithms and Complexity. — World Scientific, 2002. — P. 189–228.

41. **Shaltiel, R.** An Introduction to Randomness Extractors / R. Shaltiel // Proceedings of the International Conference on Automata, Languages and Programming. — 2011. — LNCS Vol. 6756, no. 2. — P. 21–41.

42. **Shen, A.** Combinatorial Proof of Muchnik's Theorem / A. Shen // Kolmogorov complexity and applications / M. Hutter, W. Merkle, P. Vitanyi, eds. — 2006. — Dagstuhl Seminar Proceedings 06051. — `http://drops.dagstuhl.de/opus/volltexte/2006/625`.

43. **Sipser, M.** A Complexity Theoretic Approach to Randomness / M. Sipser // Proceedings of the 15th Annual ACM Symposium on Theory of Computing. — 1983. — P. 330–335.

44. **Slepian, D.** Noiseless Coding of Correlated Information Sources / D. Slepian, J. K. Wolf // IEEE Transactions on Information Theory. — 1973. — Vol. 19. — P. 471–480.

45. **Solomonoff, R.J.** A Formal Theory of Inductive Inference, Part 1, Part 2 / R.J. Solomonoff // Information and Control (now Infromation and Computation). — 1964. — Vol. 7. — P. 1–22, 224–254.

46. **Srinivasan, A.** Computing with Very Weak Random Sources / A. Srinivasan, D. Zuckerman // SIAM Journal on Computing. — 1999. — Vol. 28. — P. 1433–1459.

47. **Sudan, M.** Decoding of Reed-Solomon Codes beyond the Error-Correcting Bound / M. Sudan // Journal of Complexity. — 1997. — Vol. 13, no. 1. — P.180–193.

48. **Ta-Shma, A.** Loss-less condensers, unbalanced expanders, and extractors / A. Ta-Shma, C. Umans, D. Zuckerman // Proceedings of the 33rd Annual ACM Symposium on the Theory of Computing. — 2001. — P. 143–152.

49. **Trevisan, L.** Construction of Extractors Using Pseudo-Random Generators / L. Trevisan // Journal of the ACM. — 2001. — Vol. 48, no. 4. — P. 860–879.

50. **Vadhan, S.** Pseudorandomness / S. Vadhan. — Draft survey/monograph. — 2012. — http://people.seas.harvard.edu/ salil/pseudorandomness/

51. **Vazirani, U.V.** "Efficient and Secure Pseudo-Random Number Generation / U.V. Vazirani, and V.V. Vazirani // Proceedings of the 25th IEEE Symposium on Foundation of Computer Science. — 1984. — P. 458–463.

52. **Viola, E.** Randomness Buys Depth for Approximate Counting / E. Viola // Proceedings of the 52nd IEEE Symposium on Foundation of Computer Science. — 2011. — P. 230–239.

53. **Zimand, M.** Two Sources are Better than One for Increasing the Kolmogorov Complexity of Infinite Sequences / M. Zimand // Proceedings of the 3rd Computer Science Symposium in Russia. — 2008. — LNCS, Vol. 5010. — P. 326–338.

54. **Zimand, M.** Extracting the Kolmogorov Complexity of Strings and Sequences from Sources with Limited Independence / M. Zimand // Proceedings the 26th Symposium on Theoretical Aspects of Computer Science. — 2009. — P. 697–708.

55. **Zimand, M.** Impossibility of Independence Amplification in Kolmogorov Complexity Theory / M. Zimand // Proceedings of the 35th International Symposium on Mathematical Foundations of Computer Science. — 2010. — LNCS, Vol. 6281. — P. 701–712.

56. **Zimand, M.** Possibilities and Impossibilities in Kolmogorov Complexity Extraction / M. Zimand // SIGACT News. — 2010. — Vol. 41, no. 4. — P, 74–94.

57. **Zimand, M.** Symmetry of Information and Bounds on Nonuniform Randomness Extraction via Kolmogorov Extractors / M. Zimand // Proceedings of the 26th IEEE Conference in Computational Complexity. — 2011. — P. 148–156.

58. **Zimand, M.** On the Optimal Compression of Sets in PSPACE / M. Zimand // Proceedings of the 18th International Symposium on Fundamentals of Computation Theory. — 2011. — P. 65–77.

Работы автора по теме диссертации

59. **Musatov, D.V.** On Extracting Space-Bounded Kolmogorov Complexity / D.V. Musatov. // Theory of Computing Systems. — 2014. — DOI 10.1007/s00224-014-9563-7.

60. **Musatov, D.V.** Space-Bounded Kolmogorov Extractors / D.V. Musatov // Proceedings of the 7th Computer Science Symposium in Russia. — 2012. — LNCS, Vol. 7353. — P. 266–277.

61. **Musatov, D.V.** Improving the Space-Bounded Version of Muchnik's Conditional Complexity Theorem via "Naive" Derandomization / D.V. Musatov // Theory of Computing Systems. — 2014. — Vol. 55, no. 2. — P. 299–312.

62. **Musatov, D.V.** Improving the Space-Bounded Version of Muchnik's Conditional Complexity Theorem via "Naive" Derandomization / D.V. Musatov // Proceedings of the 6th Computer Science Symposium in Russia. — 2011. — LNCS, Vol. 6651. — P. 64–76.

63. **Musatov, D.V.** Variations on Muchnik's Conditional Complexity Theorem / D.V. Musatov, A.E. Romashchenko, A. Shen // Theory of Computing Systems. — 2011. — Vol. 49, no. 2. — P. 227–245.

    Диссертанту принадлежат доказательства всех теорем из раздела 3.

64. **Musatov, D.V.** Variations on Muchnik's Conditional Complexity Theorem / D.V. Musatov, A.E. Romashchenko, A. Shen // Proceedings of the 4th Computer Science Symposium in Russia. — 2009. — LNCS, Vol. 5675. — P. 250–262.

    Диссертанту принадлежат доказательства всех теорем из раздела 3.

65. **Мусатов Д.В.** Упрощённое доказательство теоремы Мучника об условной колмогоровской сложности с ограничением на память / Д.В. Мусатов // Труды 55-ой научной конференции МФТИ. — М.–Долгопрудный–Жуковский: МФТИ, 2012. — С. 30–31.