

# Односторонние функции

Леонид А. Левин

Institut Des Hautes Etudes Scientifiques (Франция)  
Boston University (США)\*

август 2002

Слаб император и вся его рать  
Снова Шалтая-Болтая собратъ.<sup>1</sup>

## Резюме

Одной из наиболее важных проблем теоретической информатики является вопрос о существовании односторонних функций. Эта статья рассматривает и уточняет ряд понятий, с ним связанных. В частности, впервые приводится явное комбинаторное построение полной односторонней функции (полнота означает, что эта функция является односторонней, если таковые вообще существуют). Основные концепции содержат неожиданно много тонкостей (частично уже упоминавшихся в литературе). Здесь предлагается некоторый единый подход.

## 1 Введение I: обращение функций.

С незапамятных времён человечеству не раз напоминалось — иногда жестоко — что сделанного не обратишь. Математики столкнулись с формальным выражением этой проблемы, едва начав серьёзно анализировать основания своей науки.

---

\*Computer Sci. dept., 111 Cummington St., Boston, MA 02215; email: Lnd@bu.edu. Описанные в этой статье исследования частично были выполнены при поддержке Clay Mathematics Institute и Национального Научного Фонда США (грант NSF CCR-9820934).

<sup>1</sup>Эти строчки английской загадки комбинируют хорошо известный читателю перевод Маршака со следующим более буквальным переводом: *Круглик-Горбик на стенке сидел, // Круглик-Горбик с треском слетел. // Слаб император, слаба его рать // Круглика-Горбика снова собратъ.* — Примечание Л. А. Левина.

## 1.1 Странная аксиома.

Более века назад Георг Кантор предложил свести всё разнообразие математических понятий к единственному понятию множества, а все математические теоремы — к единственной схеме аксиом, которую можно назвать *постулатом Кантора*. Этот постулат утверждает (для каждой формулы  $A(x)$  в языке теории множеств), что существует множество, которое состоит из всех  $x$ , для которых выполнено  $A(x)$ . Это безобидное утверждение (почти что определение понятия множества), как вскоре выяснилось, имеет множество следствий, и даже больше, чем хотелось бы: из него можно вывести противоречие. Чтобы спасти положение, Цермело, Френкель и другие математики прагматически ограничили постулат Кантора некоторыми его частными случаями, разрешив лишь формулы  $A$  специального вида. При этом, вроде бы, противоречия не получается, а всё ценное продолжает выводиться. Получилась аксиоматическая теория множеств, играющая центральную роль в основаниях математики.

В 1904 году Цермело заметил, что в доказательствах математических теорем используется ещё одна аксиома, (печально) знаменитая *аксиома выбора*. Её можно сформулировать так: для всякой функции  $f$  существует обратная к ней, то есть такая функция  $g$ , для которой  $f(g(x)) = x$  при всех  $x$  из области значений функции  $f$ . Математики постепенно приняли её, хотя и неохотно: до сих пор использование этой аксиомы отмечается особо. Эта аксиома не является частным случаем постулата Кантора, к тому же имеет парадоксальные следствия. Вот простой пример.

Рассмотрим аддитивную группу  $\mathbb{T} = \mathbb{R}/\mathbb{Z}$  вещественных чисел по модулю 1, естественно отождествляемую с окружностью и с промежутком  $[0, 1)$ . Пусть  $\mathbb{Q}_{10}$  — её подгруппа, состоящая из всех конечных десятичных дробей, то есть чисел вида  $a/10^b$ . Пусть  $f$  сопоставляет с каждым числом  $x \in \mathbb{T}$  его смежный класс  $x + \mathbb{Q}_{10}$  в факторгруппе  $\mathbb{T}/\mathbb{Q}_{10}$ .

Любая обратная к  $f$  функция  $g$  выбирает в каждом смежном классе по представителю. Пусть  $G = g(f(\mathbb{T}))$  — множество этих представителей, то есть множество значений функции  $g$ . В этом случае для каждого  $x \in \mathbb{T}$  существует единственный рациональный сдвиг, переводящий  $x$  внутрь  $G$  (и  $\mathbb{T}$  разбивается на счётное число классов, получающихся из  $G$  сдвигом на элементы  $\mathbb{Q}_{10}$ ).

Мы несколько отступим от традиционного изложения, чтобы подчеркнуть элементарный характер рассматриваемого парадокса. Ещё одно (последнее) обозначение: для любого числа  $q \in \mathbb{Q}_{10}$  через  $q'$  мы обозначим число  $(10q \bmod 1)$ , получаемое отбрасыванием старшего разряда.

Пусть  $p, q$  — элементы  $\mathbb{Q}_{10}$ . Я предлагаю такое пари: если  $x + p \in G$  (для случайно выбранной точки  $x$  на окружности), то вы платите мне рубль, а если  $x + q \in G$ , то я плачу вам два. Это выгодно для вас, поскольку моя ставка больше, а условия выигрыша совершенно симметричны (соответствующие множества отличаются поворотом). Мало того, я готов быть ещё более щедрым и одновременно заключить много таких пари, по одному для каждого  $q \in \mathbb{Q}_{10}, q > 0$ ; я ставлю на  $p = q'$  против  $q$ . Как только вы соглашаетесь, мы выбираем случайное  $x \in \mathbb{T}$  (каждая его двоичная цифра получается бросанием честной монеты) и подсчитываем наши прибыли и убытки. Пусть  $\mathbf{q}$  — тот

единственный элемент множества  $\mathbf{Q}_{10}$ , для которого  $x + \mathbf{q} \in G$ . Тогда я проигрываю пари для этого  $\mathbf{q}$  и выигрываю десять пари (для всех  $q$ , при которых  $q' = \mathbf{q}$ ). Щедрость окупается, не правда ли?

Не так просто объяснить, в чём «корень зла» в этом парадоксе. В каждой игре срабатывают лишь 11 пари, и о «финансовой пирамиде» речь не идёт. Более того, если брать в качестве  $x$  случайную точку сферы, то даже общее число всех возможных пари можно сделать конечным: Банах и Тарский [1] показали, что можно построить шесть пар  $(A_i, T_i)$ , где  $A_i$  есть некоторое подмножество сферы  $S^2$ , а  $T_i$  — поворот этой сферы, с таким свойством: ставя на  $x \in A_i$  против  $T_i(x) \in A_i$  одновременно при всех  $i = 1, \dots, 6$ , мы проигрываем одно пари и выигрываем два (при любом  $x$ ). Возвращаясь к нашему примеру, отметим, что числа  $x + p$  и  $x + q$  проверяются на принадлежность к одному и тому же множеству  $G$  и отличаются в конечном числе десятичных разрядов (причём все разряды независимы и равновероятны).

Позволительно усомниться вообще в законности мысленного эксперимента, использующего бесконечное число случайных бросаний, или в осмысленности вопроса о принадлежности  $x + q$  множеству  $G$ . Но при этом мы покушаемся на самые основы теории множеств. Проще интерпретировать отказ от кажущегося выгодным пари как скрытое недоверие к аксиоме выбора.

## 1.2 Конечные объекты и полный перебор.

Теорию вычислений эти трудности с обращением функций затрагивают меньше, поскольку конечные объекты и так вполне упорядочены (принципом математической индукции) и аксиома выбора ни к чему. (Тем не менее вопрос об *эффективном обращении* функции остаётся актуальным.) К тому же шенноновская теория информации для любой функции  $f$  приписывает случайной величине  $x$  столько же информации о случайной величине  $f(x)$ , сколько  $f(x)$  об  $x$ . Колмогоровская (алгоритмическая) теория информации позволяет определить понятие количества информации для конечных (не обязательно случайных) объектов:  $I(x : y)$ , количество информации в  $x$  об  $y$ , есть разность между длинами кратчайших программ, порождающих  $y$  без использования  $x$  и с использованием  $x$ . Колмогоров и я доказали в 1967 году, что эта величина, подобно шенноновской, симметрична:  $I(x : y) \approx I(y : x)$ , хотя это равенство лишь приближённое [38].

В доказательстве есть подвох: используется полный перебор всех строк данной длины, требующий невообразимо огромного времени. Например, произведение двух простых чисел содержит всю информацию об этих числах (и наоборот), но извлечь эту информацию на практике невозможно — именно на этом предположении основана система RSA [30] и множество других приёмов современной криптографии. Колмогоров в своё время указывал на свойство симметрии информации как на одну из задач, хорошо подходящих для попытки доказать, что полный перебор неустраим ( $P \neq NP$ , как сказали бы сейчас)[18].

Появление системы RSA изменило взгляд на трудно обратимые функции: досадное препятствие стало незаменимым инструментом. За RSA последовало множество других удивительных приложений. Оказалось (и это существенно для многих из них), что

трудность обращения функции можно сконцентрировать в единственном бите. Говорят, что (легко вычисляемый) предикат  $b(x)$  является *трудным битом* (hard-core bit) для трудно обратимой (односторонней, one-way) функции  $f(x)$ , если восстановить значение  $b(x)$  по  $f(x)$  (или угадать со сколько-нибудь заметной корреляцией) столь же сложно, как восстановить всё  $x$ .

Впервые такой предикат был предложен в работе [4] для функции  $f(x) = a^x \bmod p$ . Вскоре были указаны трудные биты для системы RSA, для функции Рабина ( $x \mapsto x^2 \bmod n$ , где  $n$  есть произведение двух простых чисел), а также для «дробящих» функций вида  $f^*(x_1, \dots, x_n) = f(x_1), \dots, f(x_n)$  [37]<sup>2</sup>; в работе [7] решен общий случай произвольной односторонней функции.

Почему односторонние функции так важны? В работах [4] и [37] показано, как с их помощью можно преобразовать короткое случайное двоичное слово (seed) в неограниченную последовательность битов, неотличимых от случайных. Если односторонняя функция  $f$  является перестановкой, то годится последовательность  $g_s(i) = b(f^i(s))$  (где  $i = 0, 1, 2, \dots$ ). Общий случай разработан в работе [13].

Таким образом барьер между случайными и детерминированными процессами был размыт, и в 1980-х годах были получены многие результаты, казавшиеся ранее невыполнимыми. Среди них общие теоремы криптографии (например, [27]), конструкции доказательств с нулевым разглашением (zero-knowledge proofs) и реализация произвольных протоколов взаимодействия не доверяющих друг другу участников (как бывает, скажем, в карточных играх) в виде игр с полной информацией [8]. Этот период можно назвать золотым веком теоретической информатики, и он стал возможным благодаря односторонним функциям.

## 2 Введение II: Экстравагантные вычислительные модели.

### 2.1 Падение RSA.

Заметим, что все перечисленные конструкции основаны на односторонних (легко вычисляемых и трудно обратимых) функциях, а существование таких функций остаётся лишь гипотезой, неоднократно подвергавшейся разного рода сомнениям.

Первый эпизод такого рода связан с самим Шамиром (Adi Shamir), одним из изобретателей системы RSA (буква S = Shamir). Он доказал в [31], что разложение на множители (на трудности которого основана RSA) может быть выполнено за полиномиальное число арифметических действий. При этом каждое действие считается за одну операцию, независимо от длины чисел (“unit-cost model”). При возведении в квадрат длина числа удваивается, и повторное возведение в квадрат быстро приводит к числам космологического размера. Одно такое число может кодировать длинный массив обычных чисел, и одной операции соответствует большой объём работы (например, проверка экспоненциального числа возможных делителей). Алгоритм

---

<sup>2</sup>Доказательство леммы об изоляции (Isolation Lemma), используемой в [37] при анализе трудности бита, можно найти в [23].

Шамира не произвёл впечатления на косных криптографов: отвергнутой оказалась вычислительная модель, игнорирующая длину чисел, а не RSA.

Другая не лишённая сходства атака на RSA происходит в настоящее время и началась с замечательного результата Питера Шора. Он показал, что можно разлагать числа на множители за полиномиальное время, используя воображаемые аналоговые вычислительные машины, названные *квантовыми компьютерами* (Quantum Computer, QC). Эти машины подсказаны (доведёнными до крайности) законами квантовой физики.

## 2.2 Квантовые компьютеры.

Квантовый компьютер состоит из  $n$  взаимодействующих элементов, называемых *кубитами* (q-bits). Каждый из них является квантовой системой; её чистые состояния представляют собой единичные векторы в двумерном комплексном пространстве  $\mathbb{C}^2$ ; две компоненты вектора представляют собой амплитуды двух булевских значений. Состояние системы в целом представляет собой вектор в тензорном произведении  $n$  двумерных пространств. В этом произведении есть базис из  $2^n$  векторов, являющихся произведениями базисных векторов сомножителей. Каждый базисный вектор, таким образом, соответствует двоичному слову длины  $n$ . Квантовый компьютер охлаждают, изолируют от внешнего мира почти идеально и помещают в базисное состояние, в котором часть битов образует входное слово, а остальные биты равны нулю. Далее производится вычисление, состоящее из последовательного применения идеально обратимых преобразований, соответствующих взаимодействию кубитов. Полученное состояние может быть суперпозицией невообразимо быстро растущего числа базисных состояний с экспоненциально малыми амплитудами. Взаимодействие с окружающей средой может вносить ошибки, поэтому вычисление должно предусматривать коррекцию этих ошибок (возможную, если ошибки редки и имеют специальный вид). В остальном мы полагаемся на законы квантовой механики, считая их выполненными с неограниченной точностью. Последнее предположение весьма существенно, поскольку амплитуды экспоненциально малы и отклонения в сотом (или даже намного более далёком) знаке после запятой могут полностью изменить ход вычисления.

В работе [32] показано, что такие компьютеры могут выполнять разложение на множители за полиномиальное время. Грубо говоря, каждая из экспоненциального числа координат проверяет делимость на соответствующее число, и амплитуды концентрируются в тех координатах, которые соответствуют делителям.

## 2.3 Маленькие трудности.

Попытка реализовать компьютеры описанного вида сталкивается с массой проблем. Скажем, трудно представить себе идеальную термоизоляцию, не говоря уже о защите от нейтрино, гравитационных волн и тому подобной экзотики; их влияние на квантовые амплитуды не обязательно удовлетворяет условиям, от которых зависит коррекция ошибок. Более того, даже и классические вычисления без

рассеяния тепла остаются гипотетическими, хотя про них говорят уже несколько десятилетий (а в воображаемых мирах с искусственными законами взаимодействия их существование даже доказано). В реальном мире, где нам доступно в основном лишь электромагнитное взаимодействие между электронами, ядрами и фотонами, такого рода схемы (производящие большие вычисления с малым тепловыделением) остаются спекулятивными. Поэтому понижение температур имеет предел, а даже и небольшое количество тепла может сильно нарушить когерентность. Более того, неконтролируемые степени свободы могут быть более опасны, чем простое нагревание: кто знает, к чему может привести их взаимодействие с тончайше скоррелированными состояниями кубитов?

## 2.4 За сотни цифр от запятой.

Всё это цветочки. Основная проблема другая: в требовании, чтобы законы квантовой механики были справедливы с той фантастической точностью (сотни, если не миллионы знаков после запятой), которая нужна для квантовых алгоритмов. Физики не знают ни одного закона, который был бы справедлив с точностью до нескольких десятков знаков. Как учит история физики, каждые несколько следующих знаков требуют новых теорий и переосмысления базисных понятий. Может, при таком уровне точности надо считать квантовые амплитуды не комплексными числами, а кватернионами, раскрашенными графами или эльфами с чёрным юмором... Я подозреваю, что физики и в законы арифметики-то с такой точностью не верят. (На самом деле мы даже знаем, что основные законы физики не могут выполняться с такой точностью, поскольку они начинают противоречить друг другу!)

И вообще, какой может быть физический смысл в 500-значном числе? Пусть мы говорим: «В этом ящике хранятся кубиты, которые содержат текст Десяти Заповедей с амплитудой, в которой три десятичные цифры, начиная с пятисотой, равны 666». Есть ли хоть какой-то физический смысл в нашем утверждении? Если состояние системы близко к базисным векторам тензорного произведения, то можно надеяться переформулировать утверждение, заменив длинную десятичную дробь на несколько коротких (которые уже можно измерять). Что-то осмысленное можно вообразить для больших систем типа лазеров или конденсированных состояний, где отдельными состояниями можно пренебречь. Но разложение на множители с помощью квантовых компьютеров существенно использует экспоненциальное число глубоко индивидуальных состояний. Мне трудно представить, чем в такой ситуации можно заменить общее и прямое представление амплитуд с невообразимой точностью, разве что сделав их более «физическими» за счёт выбора менее физического базиса. Рассмотрим этот подход более подробно.

## 2.5 Как мала Вселенная.

Сторонники квантовых компьютеров часто говорят, что так или иначе мы будем в выигрыше: удастся либо сделать работающий квантовый компьютер, либо уточнить законы квантовой механики. Например, Питер Шор в [33] говорит:

Если в квантовой механике имеются нелинейные эффекты, которые можно обнаружить, наблюдая за неисправной работой квантовых компьютеров, физики весьма заинтересуются ими (я бы ожидал Нобелевской премии за убедительное свидетельство таких эффектов).<sup>3</sup>

Рассмотрим, однако, другой вариант: нам удаётся сделать квантовые компьютеры из нескольких кубитов, работающие в соответствии с теорией. С ростом числа кубитов, однако, такие компьютеры сделать всё сложнее, и прогресс останавливается задолго до того, как квантовые вычисления способны соперничать с вычислениями на обороте старого конверта. Поэтому разлагать большие числа на множители мы не можем. Можно ли хотя бы претендовать на более или менее нобелевскую премию за поправки к квантовой механике?

Однако награждающий комитет хочет увидеть что-то более специфическое, чем просто неработающий компьютер — например, явное указание состояния, в котором он находится. Но всех ресурсов вселенной оказывается недостаточно, чтобы измерить необходимое число координат с необходимой точностью. (Остаётся лишь надеяться на Нобелевскую премию по экономике, если удастся собрать деньги, необходимые для разложения пятизначных чисел на множители с помощью квантовых компьютеров!)

Сделаем некоторые оценки, обозначая через  $\sim n$  маленькие степени числа  $n$ . У криптографов числа, подлежащие разложению на множители, могут иметь тысячи (а легко возможны и миллионы) битов. Пространство состояний  $H$  размерности  $2^n$ , имеет примерно  $2^{\sim n}$  почти ортогональных друг другу векторов. Рассмотрим элемент  $v \in H$  общего положения (будем называть такие векторы «мегасостояниями»). Минимальный размер машины, которая может породить или распознать  $v$ , имеет порядок  $K = 2^{\sim n}$  — значительно больше, чем Вселенная. В самом деле, достаточно посчитать машины из  $K$  атомов: их примерно  $2^{\sim K}$ .

Есть принципиальная разница между ещё не наблюдавшимися и принципиально ненаблюдаемыми вещами. Утверждения про отдельные мегасостояния ненаблюдаемы. Может представить себе способ отличить два мегасостояния *друг от друга*, но — как показывают сделанные оценки — нет возможности отделить данное состояние (общего положения) от всех других с той степенью точности, которая существенна для поведения квантового компьютера. Какой, в таком случае, эксперимент может подтвердить правильное состояние квантового компьютера? (Даже мысленный — в предположении, что мы пользуемся ресурсами всей Вселенной, *но не более!*)

Ещё Архимед открыл, что цифровая форма представления чисел экспоненциально эффективнее аналоговой (кучи песка). Впоследствии различные аналоговые устройства редко оказывались впечатляющими. Неясно, почему квантовые компьютеры должны быть исключением.

## 2.6 Метрика или топология?

Говоря о приближениях к состояниям системы, мы обнаруживаем, что в квантовой механике отсутствует адекватный формализм. Есть тонкое различие между

---

<sup>3</sup>If there are non-linearities in quantum mechanics which are detectable by watching quantum computers fail, physicists will be VERY interested (I would expect a Nobel prize for conclusive evidence of this).

приближениями в смысле метрики и топологии. Метрика говорит о том, насколько одно (хорошее) состояние близко к специфическому другому (плохому). Топология имеет дело с близостью хорошего состояния к сочетанию всех неприемлемых (не-окрестных) состояний, и это не обязательно то же самое, что расстояние до ближайшего плохого состояния, особенно для квантовых систем.

В бесконечномерных пространствах есть разные способы отличать конечную разделённость точки и множества от нулевой (разные топологии). В конечномерном случае с формальной точки зрения такого различия нет: все топологии совпадают. Однако числа порядка  $2^{500}$  можно считать конечными лишь в весьма философском смысле, и было бы желательно ввести некое понятие слабо-топологической *глубины* окрестности, полиномиально связанное с ресурсами, необходимыми для достижения этой глубины. При этом точность, соответствующая разумной глубине, должна быть физически достижимой (возможно порождать точки внутри соответствующей окрестности, отделять центр окрестности от точек, в ней не лежащих, и т. д.).

Имея функцию расстояния, мы можем говорить об  $\varepsilon$ -окрестностях (что невозможно в топологии, где конкретное значение  $\varepsilon$  отсутствует, известно лишь существование некоторого  $\varepsilon > 0$ ). Однако  $\varepsilon$ -окрестности в метрических пространствах неизбежно обладают таким свойством (которое можно назвать «аксиомой пересечения»): пересечение любого множества  $\varepsilon$ -окрестностей (при данном  $\varepsilon$ ) есть  $\varepsilon$ -окрестность. Говоря о близости в пространстве состояний квантовой системы, мы может захотеть измерять глубину окрестности числом, не подчиняясь аксиоме пересечения. Вот пример такого рода измерения (не претендующий на пригодность для наших целей). Предположим, что окрестность нуля задаётся системой линейных неравенств  $f_i(x) < 1$ ; тогда её глубиной считается число  $1/\sum_i \|f_i\|$ , хотя ограничение точки  $x$  сферой с гильбертовой нормой 1 сделало бы эту глубину квадратично связанной с радиусом окрестности.

Возможно, что более осмысленный с точки зрения физики подход можно получить, если учитывать наличие специального базиса (составленного из тензорных произведений).

## 2.7 Не продешевить!

Разложение на множители с помощью квантовых компьютеров кажется мне научной фантастикой. Можно лишь сожалеть, что в популярных обзорах оно не отделяется от более реалистичных предложений — квантовой криптографии, квантовой информации, а также квантовых вычислений, в которых существен нелокальный доступ к информации (например, быстрый поиск в больших массивах).

Следует отметить также, что фундаментальные причины, препятствующие работе квантовых компьютеров, определённо не ясны и заслуживают серьёзного изучения. Польза от него может быть больше, чем вся мыслимая польза от быстрого разложения на множители. Представьте себе, что знаменитый демон Максвелла в своё время рекламировался бы как перспективный способ получения электроэнергии из окружающего тепла! Вероятно, в таком случае современная термодинамика появилась бы заметно позже.



В оставшейся части статьи мы не рассматриваем экстравагантные модели вычислений и твёрдо придерживаемся «полиномиального тезиса Чёрча – Тьюринга»: любое вычисление, требующее  $t$  тактов работы  $s$ -битового устройства, можно моделировать на машине Тьюринга за  $s^{O(1)}t$  шагов с памятью  $s^{O(1)}$ .

### 3 Подвохи усреднения.

Сама по себе трудность обращения функции в худшем случае (на самом трудном входе) ещё мало что значит. Представим себе, например, что все входы делятся на «лёгкие» и «трудные». Лёгкие входы  $x$  требуют времени  $\|x\|^2$  для обработки; а трудные — экспоненциального среднего времени *как для обработки, так и для нахождения* вероятностными алгоритмами. В таком случае Вселенная слишком мала, чтобы породить пример, который мы не можем решить, и обращение функций не представляет трудностей на практике.

На практике важна трудность обращения в «типичных случаях». Именно она мешает разработчикам алгоритмов и делает возможными разные криптографические чудеса. Но корректное определение «типичности» — вещь довольно тонкая.

#### 3.1 Las Vegas-алгоритмы.

Прежде всего надо договориться о способах измерения «эффективности» алгоритма обращения. Алгоритм обращения  $A(x, \alpha)$  получает на вход значение  $x = f(w)$  обрабатываемой функции  $f$  на некотором искомом  $w$ , а также последовательность случайных битов  $\alpha \in \{0, 1\}^{\mathbb{N}}$ . Если речь идёт об обращении, когда мы можем проверить ответ подстановкой, нам нет необходимости заботиться о неверных ответах, и можно предполагать, что алгоритм либо даёт верный ответ (один из прообразов элемента  $x$ ), либо не даёт никакого ответа (возможно, никогда не останавливается). Такие алгоритмы называют Las Vegas-алгоритмами.

Эффективность работы алгоритма на данном примере измеряется двумя параметрами: *объёмом*  $V_\alpha$  вычисления<sup>4</sup> (этот объём представляет собой случайную величину, функцию от  $\alpha$ ), и *вероятностью*  $p$  успешного обращения. Эти параметры связаны друг с другом: можно увеличить вероятность успеха  $p \ll 1$  за счёт повторения  $A$  при разных независимых  $\alpha$ ; при этом отношение  $p/V$  меняется мало. Поэтому кажется разумным (как это часто и делается) фиксировать значение  $p$  и требовать, чтобы алгоритм выдавал верный ответ, скажем, с вероятностью  $1/2$  или больше. Однако при этом возникает трудность: оценка вероятности  $p$  и необходимого числа повторений может потребовать экспоненциального объёма (в худшем случае), и потому имеет смысл говорить лишь о среднем объёме вычислений (для фиксированной вероятности успеха). Подчеркнём, что усреднение выполняется лишь по внутренним случайным битам ( $\alpha$ ), вход  $x$  выбирается противником. При таком подходе два параметра для измерения эффективности излишни: увеличения вероятности можно

---

<sup>4</sup>Мы говорим «объём», а не «время», имея в виду возможные модели вычислений с неограниченным параллелизмом.

достичь за счёт увеличения среднего объёма и наоборот (чтобы уменьшить средний объём, достаточно выполнять  $A$  лишь с некоторой малой вероятностью).

Можно пойти другим путём и нормализовать не вероятность успеха, а средний объём вычислений. При этом мы избегаем накладных расходов на оценку вероятности успеха и уменьшения модулярности из-за комбинирования нескольких применений алгоритма. Есть, однако, более важные аргументы в пользу такого подхода: представим себе, например, что различные прообразы данного значения односторонней функции имеют разное (и трудно сравнимое) «качество». Ограничения же среднего объёма вычисления имеют вполне ясный смысл. Для конкретной модели вычислений можно фиксировать  $O(1)$ -границу для объёма; если же мы хотим работать с различными разумными моделями, можно ограничивать средний объём вычислений многочленом от длины входа (при этом для разных алгоритмов можно брать разные многочлены). Тут, однако, возникает препятствие: множество алгоритмов с данным ограничением на средний объём вычислений не является рекурсивно перечислимым. Преодолеть эту трудность можно с помощью следующего подхода, гарантированно ограничивающего средний объём вычислений.

**Определение 1** Las Vegas-алгоритм  $A(x, \alpha)$  из класса  $LV(b)$  (в этом обозначении буква « $V$ » одновременно символизирует “Vegas” и “volume” [объём]) начинает работу, зная заранее границу  $b(x)$  на допустимый объём вычислений. При этом в любой момент алгоритм имеет право поставить «на кон» любую часть оставшегося объёма; если он проигрывает (что определяется следующим случайным битом), то эта часть пропадает, если же он выигрывает, то она удваивается. Обычно достаточно рассматривать класс  $LV(O(1))$ , который мы обозначаем просто  $L$ .<sup>5</sup>

Несмотря на жёсткое  $O(1)$ -ограничение,  $L$ -алгоритмы достаточно представительны: любой Las Vegas-алгоритм можно привести к виду из  $L$ , сохраняя (в основном) отношение между сложностью и вероятностью успеха. Если  $p$  — вероятность успеха для  $L$ -алгоритма, величина  $1/p$  соответствует числу повторений, необходимому для достижения успеха с фиксированной вероятностью, и потому играет роль времени работы алгоритма. При этом читатель, не желающий вдаваться в детали внутреннего устройства компьютеров, может принять  $L$ -алгоритмы как некую данность, и весь дальнейший анализ проводить исключительно в терминах теории вероятностей!

## 3.2 Оценка времени в терминах мультимедиан.

Нам осталось разобраться с более сложной проблемой — усреднением по возможным входам  $x$ . Определение сложности как среднего значения  $E_x t(x)$  (для алгоритма, делающего  $t(x)$  шагов на входе  $x$ ) весьма неустойчиво. В самом деле, при переходе

---

<sup>5</sup>Этот « $L$ » можно произносить как «Las», имея в виду Las Vegas, а также Laszlo Babai (который придумал название «Las Vegas-алгоритмы»). Подчеркнём, что речь не идёт об определении очередного сложностного класса: мы описываем некоторую определённую форму алгоритмов, а не класс алгоритмов или (что ещё более абстрактно) класс задач, разрешимых с помощью алгоритмов некоторого класса. Отметим, что по существу требование  $L$  не является особенно новым: оно представляет собой лишь небольшое усиление общей идеи Las Vegas-алгоритма.

от одной вычислительной модели к другой время работы может возвестись в квадрат. Такова ситуация, например, с распознаванием симметрии входа: на одноленточной машине оно требует квадратичного времени, а на двухленточной достаточно линейного. Представим себе, что подобная ситуация имеет место для значительно более медленных алгоритмов. Пусть, например, время работы алгоритма равно  $\|x\|^2$  для всех входов, кроме состоящих из одних нулей, а для  $x = 0^n$  время работы  $t(x)$  равно  $2^n$  в одной модели вычислений и  $4^n$  в другой. Будем считать, что все  $2^n$  входов данной длины  $n$  равновероятны. Тогда среднее значение  $\mathbf{E}_x t(x)$  будет полиномиальным в одной модели и экспоненциальным в другой: усреднение не коммутирует с возведением в квадрат. К тому же эта экспоненциальная оценка сложности на среднем входе не имеет практической ценности, поскольку вероятность появления сложного входа пренебрежимо мала.

Более инвариантной мерой сложности вычисления могла бы служить *медиана* времени вычислений на случайном входе, то есть минимальное число шагов, достаточное для обработки любого из *более сложной половины* входов. Эта мера, однако, неустойчива в другом смысле: она может кардинально измениться, если *половину* наиболее сложных входов заменить, скажем, на *четверть*.

К счастью, наше соглашение о Las Vegas-алгоритмах заодно решает и эту проблему: для произвольного L-алгоритма мы можем измерить вероятность успешного обращения на случайном входе (имеющем заданное распределение вероятностей). Обратная величина к этой вероятности (как функция от, скажем, размера входа) является разумной мерой *надёжности* односторонней функции. Такая мера хороша для криптографии, где ставится цель предотвратить взлом шифра даже и с малой вероятностью. В задачах, где требуется достичь успеха на почти всех входах, необходим другой подход.

**Определение 2** Пусть задано некоторое L-распределение на входах (говоря о L-распределениях, мы имеем в виду распределения на выходах L-алгоритма с пустым входом).<sup>6</sup> Порождаем вход (по этому распределению)  $k$  раз, что требует среднего времени  $O(k)$ , а затем применяем алгоритм обращения, пока не будут найдены прообразы у всех входов, для которых они существуют.<sup>7</sup> Число попыток является случайной величиной (зависящей от случайных битов, используемых в алгоритме обращения, а также от случайных битов, использованных при порождении входов). Её медиана  $\text{MT}(k)$  называется *мультимедианой* времени обращения  $f$  с помощью алгоритма  $A$ .

Эта мера качества алгоритма имеет ряд достоинств. Она коммутирует с возведением в квадрат времени работы и потому устойчива относительно перехода к другой модели вычислений. Выбор границы  $1/2$ , подразумеваемой медианой, также не является существенным. В самом деле, увеличение  $k$  в  $c$  раз соответствует такому же увеличению  $\text{MT}(k)$ , как уменьшение вероятности неудачи обращения до  $2^{-c}$ .

<sup>6</sup>Если распределение на входах не алгоритмическое, можно изменить определение, заменив в определении класса L сложность на длину выхода. При этом входы длины  $n$  должны иметь вероятности с суммой  $n^{-O(1)}$ , например,  $\theta(1)/(n \log n)^2$ .

<sup>7</sup>Входы, для которых прообразов нет, не учитываются.

Величина  $MT$  осмысленна и для верхних, и для нижних оценок. Пусть  $T(x)$  велико для  $\varepsilon$ -доли входов  $x \in \{0, 1\}^n$ . Тогда  $MT(k)$  столь же велико для  $k = n^3/\varepsilon$ . Обратно, пусть  $MT(k)$  велико. Тогда почти наверняка  $T(x_i)$  столь же велико для некоторых из  $n = k^2$  случайных входов  $x_1, \dots, x_n$  (и  $\sum_i \|x_i\| = O(n)$ ).

### 3.3 Простые распределения.

До сих пор мы учитывали влияние случайных битов в вероятностном алгоритме на время работы (для фиксированного входа), а также усредняли это время по различным входам с фиксированным распределением вероятностей. Сейчас мы обсудим выбор этого распределения, который далеко не всегда прост и отнюдь не исчерпывается ссылкой на «равномерное распределение». Такие ссылки часто лишь запутывают дело, поскольку различные распределения могут не без оснований считаться «равномерными».

Например, рассмотрим графы  $G = (V, E \subset V^2)$  с  $n$  вершинами ( $\|V\| = n$ ), где значение  $n$  выбирается с вероятностью  $\theta(1)/n^2$ . На этом множестве (при данном  $n$ ) рассмотрим два распределения, которые заслуживают названия «равномерных». Первое из них, которое мы обозначим  $\mu_1$ , случайно и равновероятно выбирает граф  $G$  среди всех  $2^{n^2}$  графов. Распределение  $\mu_2$  соответствует случайному равномерному выбору числа рёбер  $k = \|E\|$  в диапазоне от 0 до  $n^2$  и затем случайному равномерному выбору  $E$  среди  $C_{n^2}^k$  возможностей. Хотя оба распределения могут быть названы равномерными, они радикально отличаются. Например, множество  $\{G : \|E\| = n^{1.5}\}$  имеет вероятность примерно  $1/n^2$  относительно  $\mu_2$ , хотя его вероятность относительно  $\mu_1$  экспоненциально мала.

В определённом смысле все «простые» распределения вероятностей можно считать равномерными. Мы приведём соответствующее рассуждение (кратко намеченное в [22]) с некоторыми добавлениями, нужными для дальнейшего.

Отождествим (как это делают в теории множеств) каждое натуральное число  $n$  с множеством меньших натуральных чисел  $\{0, 1, 2, \dots, n-1\}$ . Мерой мы будем называть аддитивную вещественно-значную функцию на *множествах* натуральных чисел. В соответствии с нашим соглашением  $\mu(n) = \mu(\{0\}) + \mu(\{1\}) + \dots + \mu(\{n-1\})$ ; тем самым  $\mu$  представляет собой монотонную *функцию распределения*. Соответствующая *плотность* распределения задаётся формулой  $\mu'(n) = \mu(n+1) - \mu(n) = \mu(\{n\})$ ; величина  $\mu'(n)$  есть вероятность одноэлементного множества  $\{n\}$  (а не  $n$  как множества меньших натуральных чисел). Через  $\mathbf{Q}_2$  обозначим множество конечных двоичных дробей  $i/2^{\|i\|} \in [\frac{1}{2}, 1)$ . Мы округляем значения функции  $\mu$  до элементов  $\mathbf{Q}_2$ , сохраняя лишь минимально необходимое число двоичных цифр (так, чтобы вероятность изменилась не более чем в константу раз).

**Определение 3** *Говорят, что функция  $\mu : \mathbb{N} \rightarrow \mathbf{Q}_2$  вполне округлена, если  $\mu(x)$  является кратчайшей двоичной дробью в интервале  $(\mu(x-1), \mu(x+1))$ , а также  $-\log \mu(\{x\}) = O(\|x\|)$ .*

Последнее условие добавлено для удобства; его можно обеспечить, смешав (монотонную) функцию  $\mu$  с каким-нибудь простым распределением.

**Лемма 1** *Всякая вычислимая функция  $\mu: \mathbb{N} \rightarrow \mathbb{Q}_2$  может быть эффективно преобразована во вполне округлённую функцию  $\mu_1$ , вычислимую с замедлением (по сравнению с  $\mu$ ) в  $\|x\|$  раз. При этом, если  $\mu$  строго возрастает (то есть  $\mu' > 0$ ), то  $\mu'_1 \geq \mu'/4$ .*

Монотонность обеспечивается сравнением  $\mu(x)$  со всеми  $\mu(y)$  для всех  $y$ , являющихся началами  $x \in \mathbb{Q}_2$ . Далее утверждение леммы достигается за счёт округления. Сначала мы округляем  $\mu(x)$  до кратчайшей двоичной дроби  $p$ , которая ближе к  $\mu(x)$ , чем к любому другому  $\mu(y)$ ; эти округлённые значения мы называем *точками*. Затем находим все *щели*, то есть ближайшие к  $p$  двоичные дроби всех двоичных длин. Затем для каждой щели (в порядке возрастания длины) находим точку, которая её заполняет при последовательных округлениях; так делается до тех пор, пока щель для  $x$  не будет найдена.

Вполне округлённые функции  $\mu$  обладают любопытным свойством: оба числа  $m(x) = \mu(x)/\mu(\{x\})$  и  $-\log \mu(\{x\}) = \|m(x)\|$  всегда целые, и потому  $\mu(x)$  есть конечная двоичная дробь, у которой целая часть нулевая, а после запятой идёт  $m(x)$ . Поэтому  $m$  распределено почти равномерно:  $2k\mu(m^{-1}(k)) \in [1, 2]$  при  $k \in m(\mathbb{N})$ . Кроме того, оно вычислимо за полиномиальное время, так же как и  $m^{-1}$  (двоичный поиск). Поэтому  $m(x)$  можно рассматривать как альтернативное представление для  $x$ , в котором распределение  $\mu$  становится достаточно равномерным.

Вообще говоря, не всегда можно ограничиться простыми распределениями на входах. Возможно, исходные данные  $r$ , использованные при построении входа  $x$ , и имели простое распределение, но сам процесс  $A$  преобразования  $r$  в  $x$  мог быть чем-то вроде односторонней функции. Мы можем предполагать, что  $A$  есть алгоритм с не слишком большим временем работы, но не что распределение вероятностей на его выходах просто. Возникающее на выходе  $A$  распределение называется *реализуемым* (samplable). В работе [14] такие распределения сводятся к равномерным, также как и рассматриваемые данным разделе, хотя и с помощью другого трюка.

## 4 Полнота.

### 4.1 Полные распределение и инверторы.

Что значит, что данная функция трудна для обращения? Это можно уточнить двояко. Можно считать функцию трудной, если трудные для обращения значения порождаются с не слишком малой вероятностью. А можно требовать большего: чтобы вероятность получения лёгкого для обращения значения была пренебрежимо мала. Существуют различные способы свести одну задачу к другой, и мы будем рассматривать первую из упомянутых задач.

Прежде всего отметим, что лемма 1 позволяет перечислить все вычислимые за время  $t(x)$  распределения, сохраняя  $t$  с точностью до линейного множителя. Сложив все такие распределения с коэффициентами, образующими сходящийся ряд (например,  $1/i^2$ ), мы получим распределение в классе  $\text{TIME}(t(x)\|x\|)$ , являющееся полным для класса  $\text{TIME}(t(x))$ . Можно было бы соединить распределения всех сложностей в одно, при

этом каждое значение порождается с тем меньшей вероятностью, чем больше сложность его порождения (как это делалось в разделе 3.1). Но мы предпочитаем иметь дело прямо с реализуемыми распределениями.

**Определение 4** *Распределения вероятностей на выходе L-алгоритмов без входа называются реализуемыми (samplable). Аналогичным образом L-алгоритм со входом задаёт реализуемое семейство распределений (вход является параметром).*

Обычно даётся менее ограничительное определение, разрешающее больший класс алгоритмов (более близкий к LV с полиномиальным ограничением) и полиномиально большие вероятности; мы рассматриваем лишь L-алгоритмы, стремясь к большей точности.

**Предложение 1** *Существует полное (наибольшее с точностью до постоянного множителя) реализуемое семейство распределений.*

В самом деле, L-алгоритмы можно перечислять, и полное реализуемое семейство распределений можно получить, выбирая случайный L-алгоритм и выполняя его. Описанный алгоритм требует в среднем времени  $O(1)$  и имеет не меньше шансов (с точностью до постоянного множителя) породить «сюрприз», чем любой другой L-алгоритм. (У LV-алгоритмов с полиномиальным ограничением вероятность «неприятного сюрприза» может быть больше, чем у описанного алгоритма, но различие не более чем полиномиально.)

Полное распределение определено лишь с точностью до ограниченного множителя, поэтому только в логарифмической целочисленной шкале оно даёт объективную меру трудности попадания в множество  $X$  при данном значении параметра  $x$ , определённую с точностью до ограниченного числа делений шкалы.

**Обозначение 1** *Через  $Kl(X/x)$  мы обозначаем  $-\log_2 p(X/x)$ , где  $p(X/x)$  есть вероятность попадания в множество  $X$  относительно полного семейства реализуемых распределений с параметром  $x$ .*

Как часто бывает, средство для атаки помогает найти и защиту. Оптимальные алгоритмы поиска, указанные в [20, 21, 3], при нашем определении получают сами собой: полный генератор трудных задач превращается в оптимальный алгоритм их решения. Напомним, что мы «конвертируем» время работы в вероятность успеха, переходя к L-алгоритмам, и измеряем их производительность (качество) этой вероятностью. Алгоритм, порождающий наибольшее реализуемое распределение (с параметром  $x$ ) имеет наибольшую (с точностью до постоянного множителя) вероятность  $1/S(f/x) = 2^{-Kl(f^{-1}(x)/x)}$  порождения решений. Величина  $s(f/x) = Kl(f^{-1}(x)/x)$  характеризует трудность конкретного примера  $x$  и может быть названа его *непреступностью* (security). Наш оптимальный генератор в среднем требует  $O(1)$  шагов на один запуск и  $S(f/x)$  запусков. Никакой другой алгоритм не может дать лучшего результата.

**Открытая проблема.** Постоянный множитель в оптимальном алгоритме обращения может быть произвольно большим. Неизвестно, можно ли ограничить этот множитель (для достаточно длинных входов) некоторой абсолютной константой, не зависящей от выбора сравниваемого с оптимальным алгоритма (скажем, числом 10).

## 4.2 Задачи обращения и односторонние функции.

Полное распределение даёт не меньшую вероятность получить трудный вход, чем любое другое. Благодаря этому в качестве трудной для обращения функции можно взять любую NP-полную функцию: все они одинаково хороши. Однако обычно хочется найти функцию, которую трудно обратить для какого-либо обычного (например, равномерного) распределения на входах. Неожиданным образом лемма 1 как раз и указывает кодирование, преобразующее заданное распределение (вычислимое за полиномиальное время) в равномерное. С его помощью любая NP-полная функция становится максимально трудной для обращения. Однако в сочетании с таким кодированием функция теряет привлекательность, так что вопрос о построении привлекательной функции, трудной для обращения при обычном распределении на входах, сохраняется.

Как известно, ни для одной функции не удалось доказать, что она трудна для обращения, хотя многие функции кажутся таковыми. В работах [22, 34, 14, 12, 35, 36] (и других) указан ряд комбинаторных и алгебраических задач, которые являются в среднем полными при равномерном распределении входов (то есть не проще любой задачи обращения с реализуемым распределением).

Однако эти результаты всё ещё не дают односторонних функций. Разница между односторонними функциями и трудными в среднем задачами обращения может быть выражена многими способами. Простейший из них состоит в том, чтобы определить одностороннюю функцию как трудную в среднем задачу обращения функции, *сохраняющей длину*. В этом случае различие между выбором случайного аргумента или случайного значения (существенное для односторонних функций) перестаёт быть важным.

В самом деле, каждому решению соответствует только одно значение, поэтому для сохраняющих длину функций вероятность появления значения функции из данного множества значений, имеющих решения, не меньше вероятности, соответствующей равномерному распределению. При этом, однако, может быть много «близнецов» — решений, соответствующих одному и тому же значению. В этом случае мы можем модифицировать функцию следующим образом. Отгадаем логарифм числа близнецов для данного решения  $w$  (пусть этот логарифм равен  $k$ ) и рассмотрим случайный элемент  $a$  универсального семейства хеш-функций  $h_a(w)$ . Будем считать выходом функции набор  $f(w), k, a, h'_a(w)$ , где  $h'$  получается из  $h$ , если оставить только  $k$  первых битов. Содержащаяся в этом выходе дополнительная информация (при правильно угаданном значении  $k$ ) близка к случайной, и потому не помогает обращению. С другой стороны, близнецы разделяются на небольшие группы, и потому количество (и вероятность при равномерном распределении) трудных значений и количество их преобразов становятся сравнимыми. Обратное утверждение также верно:

**Предложение 2** Любая односторонняя функция с мультимедианой  $V(k)$  времени оптимального обращения (то есть неприступности  $S(x)$ ; мы предполагаем, что  $x = f(w)$  для равномерно распределённого  $w$ ) может быть преобразована в сохраняющую длину одностороннюю функцию, у которой для  $1/O(k)$ -доли примеров надёжность полиномиально связана с  $V$ .

Прежде всего, мы увеличиваем долю трудных для обращения значений, как описано в конце раздела 3.2. Если число трудных значений существенно меньше, чем число их прообразов, функцию всё равно можно переделать в сохраняющую длину без изменения трудности. Сначала мы разделяем близнецов, как описано в предыдущем абзаце. Далее, длинные значения отображаются с помощью хеш-функции в строки той же длины, что и решения. Более подробно процесс применения хеширования к односторонним функциям исследован в [14].

### 4.3 Полная односторонняя функция: продолжение замощения

Мы сейчас покажем, как модифицировать задачу о замощении (Tiling Problem), получив из неё комбинаторную полную одностороннюю функцию. Такие функции не встречались в литературе, хотя в [23] приводится построение искусственной полной односторонней функции. Конечно, хорошо бы найти несколько примеров полных односторонних функций, менее искусственных, то есть выглядящих интуитивно для человека, не знакомого (и не желающего знакомиться) с теорией алгоритмов. Мы сейчас приведём один такой пример «для затравки», надеясь на то, что рано или поздно будет накоплена «критическая масса» полных функций, сведёние которых позволит доказывать полноту разнообразных интересных кандидатов в односторонние функции.

**NP-полнота и OWF-полнота** Широкий успех доказательств NP-полноты многочисленных комбинаторных задач остаётся загадочным. Это вопрос скорее искусства, чем науки, и потому не требующий однозначного объяснения. Но одной из причин, видимо, является большой набор готовых NP-полных комбинаторных задач, описание которых не требует анализа (утомительных) деталей, характерных для *детерминированных* вычислительных моделей. Для *полноты в среднем* таких примеров пока существенно меньше, хотя они и накапливаются. С другой стороны, мой вопрос о построении хотя бы одного явного и простого примера полной односторонней функции оставался безответным в течение двух десятилетий.

Полная односторонняя функция может быть получена модификацией универсальной машины Тьюринга (Universal Turing Machine, UTM). В свою очередь, протоколы работы UTM могут быть легко преобразованы в комбинаторные объекты с определёнными свойствами (типа замощений). Описание этих свойств проще описания машины Тьюринга, поскольку теперь нет необходимости заботиться о детерминированности вычисления: отношение, которое мы строим, и не должно быть детерминированным. Упрощённые аналоги вычислений привлекательны своей простой комбинаторной структурой, которая позволяет свести возникающую задачу ко множеству других (тем самым доказав полноту последних).



Этот подход действительно позволяет построить NP-задачи, являющиеся полными в среднем (average-complete). Однако он не позволяет обеспечить условие сохранения длины, которое существенно при построении полных односторонних функций. (Условие сохранения длины может быть заменено другими требованиями, но и эти требования не удаётся удовлетворить в описанной выше конструкции.) Сейчас мы покажем, как можно сравнительно простыми средствами (понятие расширения) попытаться преодолеть возникающие проблемы. При этом мы обеспечим сохранение длины и простую комбинаторную структуру замощения. Мы надеемся, что эта полная односторонняя функция может быть полезна как исходная точка для доказательств полноты интересных односторонних функций с помощью сведений.

---

Плитки: единичные квадраты, углы которых помечены буквами; их можно прикладывать сторона к стороне, если буквы совпадают. Расширение: максимальное продолжение заданного частичного замощения квадрата с отмеченной границей в таком порядке, при котором каждая следующая плитка определяется однозначно (при заданном наборе плиток)

---

a	x	x	c
e	r	r	z
e	r	r	z
n	s	s	z

**Определение 5** Расширением замощения (*Tiling Expansion*) мы называем следующую функцию: аргумент — верхняя строка замощения и набор разрешённых плиток; значение — нижняя строка замощения, получаемого расширением верхней строки, и набор разрешённых плиток.

**Теорема 1** Функция расширения замощения является односторонней тогда и только тогда, когда односторонние функции существуют.

**Сведение:** Мы начинаем с универсальной машины Тьюринга (UTM) и добавляем к ней счётчик, который прерывает её работу после, скажем,  $n^2$  шагов. Мы сохраняем копию программы (начальный отрезок входа) неизменной, а также принудительно делаем длину выхода равной длине входа. Эта конструкция даёт нам полную одностороннюю функцию, сохраняющую длину (правда, описываемую с помощью вычислительной модели). Далее мы сводим вычисление UTM (с указанными модификациями) к задаче о замощении с помощью стандартного приёма. Мы добавляем специальный *граничный символ* и разрешаем его лишь в плитках, в которых он сочетается с символами входного или выходного алфавитов (одинакового размера), а также с *символом конца ленты* или с символом, начинающим вычисление (в зависимости от стороны плитки). Остаётся воспользоваться определением расширения.

Замощение является простой комбинаторной задачей, но её недетерминированная природа вынуждает нас указывать все плитки в квадрате, что мешает сохранению длины. Если требовать, чтобы правила замощения вынуждали детерминизм вычисления, получится громоздкая конструкция, которую трудно связать с простыми комбинаторными задачами. Вместо этого, говоря о расширениях, мы не накладываем ограничений на множество разрешённых плиток, зато разрешаем прикладывать лишь плитки, которые однозначно определяются (при данном наборе разрешённых плиток)

уже имеющимися. При этом некоторые частичные замощения квадрата удаётся продолжить до полных, добавляя плитки одну за другой, другие — нет. Этот процесс приводит к потере эффективности (небольшой для параллельных моделей), но это для нас не важно.

Остаётся интересная задача: свести эту одностороннюю функцию к другим простым комбинаторным или алгебраическим функциям, тем самым доказав их полноту.

Произвольные односторонние функции не так просто применить на практике. Во многих случаях (например, при построении псевдослучайных последовательностей) других предположений формально не требуется, но приходится использовать построения, которые катастрофически ухудшают количественные показатели эффективности. Более пригодные на практике конструкции используют односторонние функции с некоторыми дополнительными свойствами, например, с малой энтропией в смысле Реньи. Это же требование используется при преобразовании слабо односторонней функции в сильно одностороннюю (с сохранением параметров надёжности, как описано в [6]). Следующее замечание описывает один из возможных путей получения односторонней функции, удовлетворяющей этому требованию. (В нём выражение  $f(x) + ax$  может быть заменено на другие хеш-функции.)

**Наблюдение 1** Аргументы функции  $g(a, x) = (a, f(x) + ax)$  в среднем имеют не более одного близнеца для любой сохраняющей длину функции  $f$  и для  $a, x \in \text{GF}_{2^{\|x\|}}$ .

**Гипотеза 1** Построенная таким образом функция  $g$  является односторонней, если функция  $f$  была таковой, и имеет тот же (с точностью до полиномиального множителя) параметр надёжности.

## References

[FOCS] *Proceedings of the Annual IEEE Symposium on Foundations of Computer Science.*

[STOC] *Proceedings of the Annual ACM Symposium on Theory of Computing.*

- [1] S. Banach, A. Tarski. Sur la decomposition des ensembles de points en parties respectivement congruentes. *Fund. Math.* 6, 244–277, 1924.
- [2] S. Ben-David, B. Chor, O. Goldreich, M. Luby. On the Theory of Average Case Complexity. [STOC], 1989, pp. 204–216.
- [3] Charles H. Bennett. Logical Depth and Physical Complexity. В кн.: Rolf Herken, ed. *The Universal Turing Machine—a Half-Century Survey*. Oxford University Press 227–257, (1988).
- [4] Manuel Blum, Silvio Micali. *How to generate cryptographically strong sequences of pseudo-random bits*. Sicomp. 13:850–864 (1984).
- [5] W. Diffie, M. E. Hellman. New Directions in Cryptography. *IEEE transactions on Info. Theory*, IT-22:644–654, 1976.

- [6] Oded Goldreich, Russell Impagliazzo, Leonid A. Levin, Ramarathnam Venkatesan, David Zuckerman. Security Preserving Amplification of Hardness. [FOCS], 1990, pp. 318–326.
- [7] Oded Goldreich, Leonid A. Levin. A Hard-Core Predicate for any One-Way Function. [STOC], 1989, pp. 25–32.
- [8] O. Goldreich, S. Micali, A. Wigderson. Proofs that Yield Nothing but their Validity. J.ACM, 38(3):691–729, 1991.
- [9] Yuri Gurevich. Average Case Complexity. *Internat. Symp. on Information Theory, Proc.*, IEEE, 1985.
- [10] Yuri Gurevich. Complete and Incomplete Randomized NP Problems. [FOCS], 1987, pp. 111–117.
- [11] Yuri Gurevich. The Challenger-Solver Game: Variations on the Theme of  $P=?NP$ ". Bull. Europ. Assoc. for Theor. Comp. Sci., Oct. 1989, pp. 112–121.
- [12] Yuri Gurevich. Matrix Decomposition is Complete for the Average Case. [FOCS], 1990.
- [13] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, Michael Luby. A Pseudorandom Generator from any One-way Function. *SICOMP* 28(4):1364–1396, 1999.
- [14] Russell Impagliazzo, Leonid A. Levin. No Better Ways to Generate Hard NP Instances than Picking Uniformly at Random. [FOCS], 1990, pp. 812–821.
- [15] David S. Johnson. The NP-Completeness Column. *J. of Algorithms* 5:284–299, 1984.
- [16] Richard Karp. The probabilistic analysis of some combinatorial search algorithms. *Algorithms and Complexity*. (J.F.Traub, ed.) Academic Press, NY 1976, pp. 1–19.
- [17] Donald E. Knuth. The Art of Computer Programming. v.2 *Seminumerical Algorithms*. Addison-Wesley, 3d ed., 1997. Sec. 3.5.F. См. также с. 10, 29–36 в файле <http://www-cs-faculty.stanford.edu/~knuth/err2-2e.ps.gz>.
- [18] А. Н. Колмогоров. Несколько теорем об алгоритмической энтропии и алгоритмическом количестве информации. Доклад на заседании Московского математического общества 31 октября 1967 г. Резюме: *Успехи математических наук*, 1968, т. 21, вып. 2. с. 201.
- [19] А. Н. Колмогоров, В. А. Успенский. Алгоритмы и случайность. *Теория вероятностей и её применения*, 1987, т. 32, вып. 3, с. 425–455.
- [20] Л. А. Левин. Универсальные задачи перебора. *Проблемы передачи информации*, 1973, т. 9, вып. 3, с. 265–266.
- [21] Leonid A. Levin. Randomness Conservation Inequalities. *Information and Control* 61(1):15–37, 1984.

- [22] Leonid A. Levin. Average Case Complete Problems. *SIAM J. Comput.* 15(1):285–286, 1986.
- [23] Leonid A. Levin. One-way functions and pseudorandom generators. *Combinatorica* 7(4):357–363, 1987.
- [24] Leonid A. Levin. Randomness and Non-determinism. *J. Symb. Logic*, 58(3):1102–1103, 1993. Та же статья, за вычетом технических деталей, опубликована в сборнике: *International Congress of Mathematicians*, Zurich, August 1994. *Proceedings* (Invited Addresses), pp. 1418–1419. Birkhauser Verlag, 1995.
- [25] G.L.Miller. Riemann’s Hypothesis and tests for Primality, *J. Comp. Sys. Sci.* 13(3):300–317, 1976.
- [26] Ming Li, Paul M.B. Vitányi. Introduction to Kolmogorov Complexity and its Applications. Springer Verlag, New York, 1993.
- [27] M. Naor, M. Yung. Universal One-way Hash Functions and Their Applications. [STOC], 1989, pp. 33–43.
- [28] Michael Rabin. *Digitalized Signatures as Intractable as Factorization*. MIT/LCS/TR-212, 1979.
- [29] M.O. Rabin. Probabilistic Algorithms for Testing Primality. *J. Number Theory*, 12:128–138, 1980.
- [30] Ronald L. Rivest, Adi Shamir, Leonard M. Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *CACM* 21(2):120–126, 1978.
- [31] Adi Shamir. Factoring Numbers in  $O(\log n)$  Arithmetic Steps. *Information Processing Letters* 8(1):28–31, 1979.
- [32] Peter W. Shor. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIComp*. 26(5):1484–1509 (1997).
- [33] Peter W. Shor. “Re: When will quantum computers become practical?” Usenet-группа *sci.physics.research*, статья FrsHL0.Ilr@research.att.com, 3/22/2000: <http://groups.google.com/groups?ic=1&selm=FrsHL0.Ilr%40research.att.com> Также доступна по адресу: <http://www.lns.cornell.edu/spr/2000-03/threads.html#0022485>.
- [34] Ramarathnam Venkatesan, Leonid A. Levin. Random Instances of a Graph Coloring Problem are Hard. [STOC], 1988.
- [35] Ramarathnam Venkatesan, Sivaramakrishnan Rajagopalan. Average case intractability of Matrix and Diophantine Problems. [STOC], 1992, pp. 632–642.
- [36] J. Wang. Average Case Completeness of a Word Problem for Groups. [STOC], 1995, pp. 325–334

- [37] Andrew C. Yao. Theory and applications of trapdoor functions. [FOCS], 1982, pp. 80–91.
- [38] А. К. Звонкин, Л. А. Левин. Сложность конечных объектов и обоснование понятий информации и случайности с помощью теории алгоритмов. *Успехи математических наук*, 1970, т. 25, вып. 6, с. 85–127.