Условная сложность и коды

Ан. А. Мучник

Аннотация

Пусть x и y — произвольные двоичные слова; мы доказываем, что существует программа p длины около $K(x|y)$, которая переводит y в x и имеет малую сложность относительно x. (Имея в виду известную аналогию между шенноновской теорией информации и колморовской сложностью, можно сказать, что этот результат родственен теоремам Вольфа – Слепяна и Кёрнера – Чисара – Мартона (см. [1].)

Кроме того, показано, что для любых трёх слов x, y, z длина кратчайшей программы, переводящей оба слова y и z в x, равна максимуму из $K(x|y)$ и $K(x|z)$ (с точностью до $O(\log n)$, если x, y, z — слова длины не более n).

1. Введение

Пусть даны два произвольных двоичных слова x и y. Условной колмогоровской сложностью $K(x|y)$ слова x относительно слова y называют длину кратчайшей программы, которая переводит y в x. Точное значение $K(x|y)$ зависит от выбора способа записи программ, однако произвол тут невелик: различные естественные определения (исходное определение Колмогорова, префиксная сложность и другие, см. [6]) отличаются не более чем на $O(\log n)$ для слов длины n. Мы будем пренебрегать разницей порядка $O(\log n)$, так что нам не важно, какое именно определение использовать.

Можно было бы полагать, что кратчайшая программа содержит ту и только ту информацию, которая есть в x и отсутствует в y. Однако ситуация не так проста: программы, переводящие y в x, могут обладать разными свойствами. В самом деле, пусть x и y — два независимо выбранных случайных слова длины n. Рассмотрим две программы, переводящие y в x. Первая из них игнорирует вход и печатает x; вторая на входе t даёт ответ $t \oplus p$, где $\oplus$ обозначает побитовое сложение по модулю 2, а p подобрано так, что $y \oplus p = x$ (другими словами, $p = x \oplus y$). Эти две программы не имеют ничего общего (в том смысле, что слова x и p, их задающие, не имеют общей информации).

Оказывается, что для любых x и y найдётся программа p, переводящая y в x, длина которой близка к минимуму (то есть к $K(x|y)$) и которая проста относительно x (то есть $K(p|x) \approx 0$). Точная формулировка и доказательство этого результата приводятся ниже. (В нашем примере этим свойством обладает первая программа, но не вторая.) Тем самым даётся частичный ответ на вопрос, поставленный в статье [2] (Remark 3.8, см. подробнее в [7]).

Метод, с помощью которого строится такая программа p, применим и в более общей ситуации, когда условий несколько. С его помощью мы доказываем, что для любых трёх слов x, y, z длины не больше n найдётся программа длины не более $\max(K(x|z), K(y|z)) + O(\log n)$, которая переводит любое из слов x и y в z. Задача о построении такой программы рассматривалась в работе Горбунова [4], который показал, что при замене дополнительного слагаемого $O(\log n)$ на $O(\log(K(x|y) + K(x|z)))$ этого сделать нельзя. Отметим также, что взяв в качестве x пару $\langle y, z\rangle$, мы получаем в качестве следствия известный результат из работы [2] о минимальной длине программы, переводящей y в z и обратно.

2. Программы и коды

Чтобы явно не упоминать способ программирования, мы будем говорить не о программах, а о кодах. Слово p мы будем называть *кодом* слова x *при известном* y, если условная сложность слова x при известных y и p близка к нулю (точная граница будет указываться в формулировках). Легко заметить, что

- если p есть код x при известном y, то сложность p (и тем более длина p) не могут быть сильно меньше $K(x|y)$;
- для любых x и y существует слово p, являющееся кодом x при известном y и имеющее длину $K(x|y)$.

Точные формулировки таковы:

- $K(p) \geq K(x|y) - K(x|y,p) - O(\log n)$ для любых слов x, y, p длины не более n;
- для любых x и y длины не более n найдётся слово p длины $K(x|y)+O(\log n)$, для которого $K(x|y,p) = O(\log n)$

В самом деле, первое неравенство становится очевидным, если переписать его так:

$$K(x|y) \leq K(x|y,p) + K(p) + O(\log n)$$

(из программы для p и программы, дающей x при известных y и p, можно составить программу, дающую x при известном y; накладные расходы на составление такой программы не превосходят $O(\log n)$).

С другой стороны, если в качестве p взять кратчайшую программу, переводящую y в x, то $|p| = K(x|y)$ по построению и $K(x|y,p) = O(\log n)$, поскольку по y и p можно получить x, применив p к y.

Заметим, наконец, что из кода для x при известном y можно получить программу примерно той же длины, которая переводит y в x. В самом деле, если p — такой код, то $K(x|y,p)$ мало, поэтому есть короткая программа, переводящая пару (y,p) в x. Фиксируя в этой программе второй аргумент равным p, мы получаем программу длины примерно $|p|$ (основную часть программы составляет как раз константа p), которая переводит y в x.

3. Коды минимальной длины

Отметим, что само слово x является кодом x при (любом) известном y. Поэтому среди всех кодов x при известном y заведомо есть коды, простые относительно x. Оказывается, что это свойство можно сочетать с требованием минимальной длины: среди кодов x при известном y есть код длины $K(x|y)$, простой относительно x.

(Заметим в скобках, что если ослабить требование и искать код минимальной длины, простой относительно пары (x,y), то это легко: будем перебирать все программы длины $K(x|y)$ и искать среди них программу, переводящую y в x; первая найденная и будет искомым кодом, так как для её задания при известных x и y достаточно указать значение $K(x|y)$, что требует $O(\log n)$ битов.)

Теорема 1. *Пусть x и y — произвольные слова длины короче n. Тогда найдётся слово p длины $K(x|y)$, для которого $K(p|x) = O(\log n)$ и $K(x|p,y) = O(\log n)$.*

(Константы в $O(\log n)$-обозначениях не зависят от n, x и y.)

4. Схема доказательства

Вначале опишем основные идеи доказательства. (Сходные методы в другой ситуации используются в [3].) Пусть X — множество всех слов длины менее n. Обозначим сложность $K(x|y)$ через m. Рассмотрим множество P всех слов длины m Мы будем применять «хеширование», для чего рассмотрим хеш-функцию $\chi\colon X \to P$. (Откуда такие функции берутся и какими свойствами они обладают, мы скажем ниже.) В качестве искомого кода слова x при известном y будем пробовать хеш-значение $p = \chi(x)$ слова x. Оно просто относительно x (если сама хеш-функция χ проста). Но как восстановить x, зная $p = \chi(x)$ и y? Будем перебирать все слова x' длины менее n, у которых $K(x'|y) \leq m$ (применяя параллельно все программы длины не более m к слову y). Множество всех таких x' мы обозначим X_y. Для каждого $x' \in X_y$ мы вычисляем $\chi(x')$.

Если нам повезёт и в X_y есть только одно слово x', для которого $\chi(x') = p$ (то есть само x), то описанный процесс рано или поздно даст нам x, так что сложность x относительно y и p мала (для восстановления x по y и p достаточно указать числа n, m и хеш-функцию χ, которую мы предполагаем простой). Но где гарантии, что такое x' только одно и как построить хеш-функцию?

5. Три модификации

Чтобы описанный план стал выполнимым, нам придётся пойти на уступки трёх видов.

Несколько хеш-функций

Во-первых, мы будем рассматривать не одну хеш-функцию χ, а семейство $\chi_1, \ldots, \chi_N$ из N хеш-функций. Число N будет ограничено полиномом от n, поэтому значение любой из выбранных хеш-функций в точке x будет просто относительно x (в предположении, что само семейство простое). В самом деле, для получения $\chi_i(x)$ достаточно указать i, для чего требуется $\log N = \log \mathrm{poly}(n) = O(\log n)$ битов. Если хотя бы одна из функций χ_i выделяет интересующее нас x в множестве X_y (то есть $\chi_i(x) \neq \chi(x')$ для любого $x' \in X_y$, отличного от x), то этого нам вполне достаточно: для восстановления x по y и $p = \chi_i(x)$ достаточно знать m, n и i (помимо самого семейства хеш-функций, которое мы предполагаем простым).

Несколько прообразов

Сделаем ещё один шаг. Заметим, что нам не обязательно, чтобы элемент $x' \in X_y$, для которого $\chi_i(x') = p$, был ровно один. Достаточно, чтобы таких элементов было не слишком много (их число должно быть ограничено полиномом от n). В самом деле, если это так, то для восстановления x по y и p нужно дополнительно указать лишь порядковый номер x в перечне всех x', у которых $\chi_i(x') = p$ (в порядке появления при перечислении).

Не все $x \in X_y$ могут быть хорошими

Теперь мы видим, что достаточно построить простое семейство хеш-функций полиномиального (по n) размера с таким свойством: для любого элемента $x \in X_y$ (напомним: это означает, что $|x| < n$ и $K(x|y) \leq m$) найдётся хеш-функция χ_i, для которой количество $x' \in X_y$ с $\chi_i(x) = \chi_i(x')$ ограничено полиномом от n. Третья и последняя уступка состоит в следующем: достаточно, чтобы это свойство было выполнено для всех $x \in X_y$, кроме сравнительно небольшой их части. В самом деле, назовём те $x \in X_y$, для которых оно не выполнено, «плохими» (а остальные — «хорошими»). Множество плохих x (тех, которые склеиваются всеми функциями χ_i с большим числом других слов из X_y) можно перечислять, зная y, m и n. Поэтому если плохих x немного, то их условная сложность относительно y будет меньше m (а по предположению $K(x|y) = m$, значит, интересующее нас x будет хорошим).

6. Откуда берутся хеш-функции

После всех этих уступок нужно убедиться, что семейство хеш-функций с интересующими нас свойствами существует. Удобно изображать это семейство в виде двудольного графа. Слева стоят слова из X, справа — слова из P (таким образом, слева имеется $2^n - 1$ слов, а справа — 2^m). Каждое слово $x \in X$ соединим N рёбрами с его хеш-значениями $\chi_1(x), \ldots, \chi_N(x)$. (Если некоторые из этих значений совпадут, в графе будут кратные рёбра.) Нас интересует часть этого графа, которая получится, если слева оставить лишь вершины из X_y при некотором y. Какие вершины из X_y будут хорошими (в описанном выше смысле)? Если у какой-то вершины $x \in X_y$ есть правый сосед p, у которого мало левых соседей (в ограниченном на X_y графе), то такая вершина заведомо хороша: p является хеш-значением, которое встречается лишь у небольшого числа элементов множества X_y. (Обратите внимание, что мы не смотрим, значением какой именно хеш-функции

является данный сосед. Равенство $\chi_i(x) = \chi_j(x')$ при $i \neq j$ нам ничем не мешает, но используемый способ подсчёта этого не учитывает.)

Итак, в графе на $X_y \times P$ мы выделяем плохие хеш-значения (вершины справа, имеющие много соседей слева). Вершина слева будет заведомо хорошей, если хотя бы один её сосед справа хорош (имеет мало соседей слева). Вершины, для которых это не так (все соседи справа плохие), мы будем называть «опасными»; нам надо, чтобы их было немного. (Как мы уже говорили, среди опасных вершин могут быть и хорошие, но мы этого не учитываем.)

Далее план таков. Пусть граф (исходный, в $X \times P$) обладает свойством типа экспандера: для всякого множества $S \subset X$ мощности несколько меньше 2^m множество S' всех соседей всех вершин из множества S содержит не меньше элементов, чем S. Это свойство, очевидно, сохранится и при ограничении графа на $X_y \times P$. Оно гарантирует, что если плохих вершин справа мало, то и опасных вершин слева мало (все соседи опасных вершин — плохие). А число плохих вершин справа можно оценить сверху так: оно не превосходит общего числа рёбер графа, делённого на степень плохой вершины справа (которая достаточно велика по определению плохой вершины). Общее число рёбер графа (ограниченного на X_y) есть $|X_y| \times N$ (из каждой из $|X_y|$ вершин слева выходит N рёбер).

Осталось сказать, откуда берётся семейство хеш-функций с требуемым свойством. Мы доказываем (вероятностно), что такие семейства существуют, а затем говорим, что для данных n и m можно проверять все наборы функций по очереди, пока не обнаружится набор с нужным свойством. Первый из таких наборов алгоритмически определяется по n и m и потому имеет сложность $O(\log n)$.

7. Оценка вероятностей

Лемма 1. *Пусть m и n — целые положительные числа, причём $m \leq n$. Пусть X и P — конечные множества мощности $2^n - 1$ и 2^m соответственно. Пусть целые положительные числа N, u таковы, что*

$$\frac{u}{2^m} < 2^{-(m+n)/N}$$

Тогда существует набор из N функций

$$\chi_1, \ldots \chi_N \colon X \to P$$

с таким свойством: для всякого множества $U \subset X$ размера u множество всех $\chi_i(u)$ при всех $u \in U$ и всех $i \in \{1, \ldots, N\}$ содержит не менее u элементов.

Доказательство леммы. Покажем, что для случайно выбранных функций $\chi_1, \ldots, \chi_N$ требуемое свойство выполнено с положительной вероятностью. В самом деле, имеется не более 2^{nu} различных множеств U: на каждом из u шагов мы можем выбрать любой из $2^n - 1$ элементов. (Эта оценка не учитывает порядок выбора и возможность повторений.)

Для фиксированного U подсчитаем вероятность того, что все значения $\chi_i(u)$ (при всех $u \in U$ и при всех i) попадают в некоторое множество $V \subset P$, содержащее $u - 1$ элементов. Эта вероятность меньше $(u/2^m)^{Nu}$ (поскольку значения N случайных функций в u различных точках независимы и для каждого из них вероятность попасть в V меньше $u/2^m$). Число различных множеств V не превосходит 2^{mu}, поэтому суммарная (по всем U и V) вероятность неблагоприятного исхода меньше

$$2^{mu} \cdot 2^{nu} \cdot \left(\frac{u}{2^m}\right)^{Nu}$$

и нужно, чтобы это число было меньше единицы. Извлекая корень степени Nu, получаем условие леммы. (Конец доказательства леммы.)

Мы будем использовать лемму при

$$\frac{u}{2^m} < 1/n.$$

Тогда достаточно взять $N = 2\lceil n/\log n \rceil$, так что для указания любой из хеш-функций достаточно $O(\log N) = O(\log n)$ битов (надо указать значения m и n, а также порядковый номер хеш-функции).

8. Оценка числа соседей

Перейдём теперь к оценкам числа плохих и опасных вершин. Напомним, что мы рассматриваем слова x и y длины меньше n, для которых $K(x|y) = m$. Через X мы обозначаем множество всех слов длины не больше n, через P — множество всех слов длины m (эти слова используются как хеш-значения).

Множество X содержит $2^n - 1$ элементов; множество P содержит 2^m элементов. Мы применяем лемму 1 к множествам X и P, положив $N = 2\lceil n/\log n\rceil$ (о значении u мы скажем позже). Получается двудольный граф на $X \times P$.

Для данных x и y мы называем хеш-значение $p \in P$ плохим, если оно имеет более n^c соседей в множестве X_y всех слов длины меньше n, имеющих сложность не более m относительно y. (Точное значение достаточно большой константы c мы выберем позднее.) Как мы видели, число плохих хеш-значений меньше

$$|X_y| \cdot N/n^c < 2 \cdot 2^m \cdot 2n/n^c = \left(\frac{4}{n^{c-1}}\right) \cdot 2^m$$

(напомним, что $|X_y| < 2 \cdot 2^m$, поскольку число всех программ длины не более m меньше $2 \cdot 2^m$, и $N < 2n$). Обозначим целую часть правой части этого неравенства через u. При $c \geq 4$ отношение $u/2^m$ меньше $1/n$, и можно применить лемму с $N = \lceil 2n/\log n\rceil$. По лемме число опасных вершин слева также меньше u (иначе u опасных вершин слева имели бы только плохих соседей, а число плохих вершин справа меньше u и свойство экспандера было бы нарушено). Следовательно, сложность любой опасной вершины при известном y не превосходит

$$O(\log n) + \log\left(\frac{4}{n^{c-1}} \cdot 2^m\right) = O(\log n) + m - (c-1)\log n$$

(для задания опасной вершины надо указать n и m, после чего определится набор хеш-функций и останется указать номер вершины в перечислении всех опасных для данного y вершин). Видно, что при достаточно больших c (на самом деле достаточно взять, скажем, $c = 4$) и достаточно больших n эта сложность меньше m и потому исходная вершина x не может быть опасной. Это значит, что у неё есть хороший сосед p. Тогда $K(p|x) = O(\log n)$ (достаточно указать хеш-функцию, дающую p по x) и $K(x|y,p) = O(\log n)$ (вершина x является одним из не более чем n^c соседей вершины p в X_y, так что для задания x при известных y и p достаточно указать n, m и порядковый номер вершины x в перечислении соседей вершины p, который ограничен полиномом от n).

Теорема 1 доказана.

Замечание. Код p, обладающий указанными в теореме свойствами, определён неоднозначно. Пусть, например, y и z — независимые случайные слова равной длины, а $x = yz$. Тогда слова z и $y \oplus z$ годятся в качестве p, но не имеют общей информации.

9. Коды для нескольких условий

Описанный метод построения кодов с помощью хеш-функций имеет и другие применения. В частности, он позволяет построить общий код для нескольких условий. Пусть, например, нам даны слова x, y, z длины менее n, причём $K(x|y) = K(x|z) = m$. Оказывается, что тогда можно найти такое слово p длины m, для которого

$$K(p|x) = O(\log n);$$
$$K(x|p,y) = O(\log n);$$
$$K(x|p,z) = O(\log n).$$

Другими словами, p является кодом x при известном y, а также кодом y при известном z. (К тому же p просто относительно x — впрочем, и без этого требования утверждение вполне содержательно.)

Доказательство этого утверждения (и его обобщения) приведены ниже. Предварительно мы опишем схему доказательства.

Как и раньше, мы рассматриваем семейство хеш-функций

$$\chi_1, \ldots, \chi_N \colon X \to P,$$

где X — множество всех слов длины менее n, а $P = \mathbb{B}^m$ — множество всех слов длины m. В качестве кода p используется значение одной из функций χ_i на элементе $x \in X$. Семейство хеш-функций задаёт двудольный граф на $X \times P$ (один и тот же для слов y и z). Однако множества X_y и X_z (состоящие их слов сложности не более m относительно y и z) различны, так что нам придётся рассматривать два ограничения этого графа: на $X_y \times P$ и на $X_z \times P$. Понятие плохой вершины в P будет различным для этих двух графов. Доказательство теоремы 1 показывает, что вершина x имеет y-хорошего соседа, а также z-хорошего соседа, но эти соседи могут быть разными, что нас не устраивает.

Эту трудность можно преодолеть так: если мы докажем, что *большинство* соседей вершины x являются y-хорошими, а также что *большинство* соседей вершины x являются z-хорошими, то тем самым найдётся сосед p, который одновременно является y-хорошим и z-хорошим. Это хеш-значение p и будет искомым кодом.

Как доказать, что большинство соседей вершины x являются y-хорошими (z-хорошими)? Нам придётся изменить определение опасной вершины. Мы будем называть вершину y-опасной (z-опасной), если доля y-хороших (z-хороших) вершин среди её соседей не превосходит $1/2$, и докажем, что данная нам вершина x является y-безопасной и z-безопасной.

Соответственно следует изменить и требование типа экспандера, накладываемое на граф в $X \times P$. Теперь оно будет таким: для всякого множества $Z \subset P$, содержащего u вершин, число вершин в X, у которых половина или более соседей попадает в Z, меньше u. (О выборе значения u мы скажем ниже.) Заметим, что это требование усиливает ранее использованное: если заменить «половина или более соседей» на «хотя бы один сосед» получится переформулировка старого требования.

Теперь сформулируем обобщение, о котором мы говорили. Пусть (для данных слов x, y, z) сложности $K(x|y)$ и $K(x|z)$ различны: $K(x|y) = a$ и $K(x|z) = b$, причём, скажем, $a > b$. Оказывается, что тогда можно найти согласованные коды: существутет слово p длины a, являющееся кодом слова x при известном y, а также слово q длины b, являющееся кодом слова x при известном z, для которых q является началом p. (Оба слова p и q к тому же просты относительно x.)

Это обобщение требует и соответствующего изменения доказательства: мы рассматриваем хеш-функции со значениями в словах длины a, а также их «усечённые варианты», которые получатся, если в хеш-значениях оставить только первые b битов. Требуется, чтобы оба семейства (полное и «урезанное») обладали указанным выше свойством типа экспандера. (Нам будет удобно усилить это требование и рассматривать хеш-функции, у которых начальные отрезки любой длины t (не только a и b) обладают этим свойством.)

Имея согласованные коды p и q с указанными свойствами, легко построить программу M, переводящую любое из слов y и z в слово x и имеющую длину

$$\max(K(x|y), K(x|z)) + O(\log n)$$

Программа M начинает с определения того, какое из слов y и z ей дано (эти слова отличаются в каком-то бите; зная его номер, можно их разделить), после чего применяет одну из двух программ длины $O(\log n)$, переводящих p, y или q, z в x. В программу M встроены слова p и q (достаточно встроить более длинное из них и указать длину более короткого, так как оно является начальным отрезком).

10. Оценка вероятностей: случай нескольких условий

Сейчас мы сформулируем обобщение леммы 1, которе используется при построении общего кода для нескольких условий. Будем обозначать через $\mathbb{B}^t$ множество всех двоичных слов длины t. Если $x \in \mathbb{B}^t$ и $s \leq t$, через $[x]_s$ мы обозначаем начало слова x, имеющее длину s.

Лемма 2. *Пусть даны натуральные числа n и N, а также положительное число ε, причём*

$$n 2^{N+2n+1} \varepsilon^{N/2} < 1.$$

Тогда существует семейство отображений

$$\chi_1, \ldots, \chi_N \colon \mathbb{B}^n \to \mathbb{B}^n$$

с таким свойством: для любого $m \in \{1, \ldots, n\}$ и для любого подмножества $Q \subset \mathbb{B}^m$, число элементов в котором не превосходит $\varepsilon 2^m$, количество тех $x \in \mathbb{B}^n$, для которых

$$[\chi_i(x)]_m \in Q \quad \textit{для половины или более значений } i \in \{1, \ldots, N\}$$

меньше $|Q|$ (числа элементов в Q).

Доказательство. Мы покажем, что для случайно выбранных функций $\chi_1, \ldots, \chi_N$ (все значения $\chi_i(x)$ при всех i и x независимы и равномерно распределены в $\mathbb{B}^n$) вероятность нарушения указанного свойства меньше единицы. Эту вероятность мы оценим сверху. Для каждого $m \le n$, для каждого $u \le \varepsilon 2^m$ и для любых множеств $P \subset \mathbb{B}^n$ и $Q \subset \mathbb{B}^m$, содержащих по u элементов, оценим вероятность того, что для каждого элемента $x \in P$ не менее половины значений $[\chi_i(x)]_m$ (при $i = 1, \ldots, n$) попадает в Q. Для фиксированного x вероятность того, что не менее половины его соседей попадает в Q, не превосходит $2^N \varepsilon^{N/2}$, поскольку для каждого из не более чем 2^N подмножеств множества $\{1, \ldots, N\}$, содержащих $\lceil N/2 \rceil$ элементов, вероятность того, что все входящие в него значения i ведут внутрь Q, не превосходит $\varepsilon^{N/2}$ (значения $[\chi_i(x)]_m$ при разных i независимы и равномерно распределены в $\mathbb{B}^m$). Такое событие должно произойти независимо для всех $x \in P$, так что полученную оценку надо возвести в степень u.

Таким образом, для интересующей нас вероятности (которая должна быть меньше единицы) мы получаем оценку

$$\sum_{m=1}^{n} \sum_{u=1}^{\varepsilon 2^m} \sum_{P \subset \mathbb{B}^n, |P|=u} \sum_{Q \subset \mathbb{B}^m, |Q|=u} (2^N \varepsilon^{N/2})^u$$

Число различных множеств P не превосходит 2^{nu} (столько есть последовательностей длины u, составленных из элементов $\mathbb{B}^n$); число различных множеств Q не превосходит 2^{mu}. Учитывая это, получаем оценку

$$\sum_{m=1}^{n} \sum_{u=1}^{\varepsilon 2^m} 2^{un}\, 2^{um}\, 2^u N\, \varepsilon^{Nu/2}$$

или

$$\sum_{m=1}^{n} \sum_{u=1}^{\varepsilon 2^m} \left(2^n\, 2^m\, 2^N\, \varepsilon^{N/2}\right)^u$$

Внутренняя сумма представляет собой геометрическую прогрессию. В условиях леммы знаменатель этой прогрессии меньше $1/2$, и потому сумма прогрессии не превосходит удвоенного первого члена, который не зависит от u. Учитывая это, получаем оценку

$$2 \cdot 2^N \cdot n \cdot (2^{n+m} \varepsilon^{N/2}) < n 2^{N+2n+1} \varepsilon^{N/2},$$

что меньше единицы по условию леммы.

Лемма 2 доказана.

Мы будем использовать эту лемму при $\varepsilon = 1/n$. В этом случае можно переписать условие леммы как

$$n 2^{N+2n+1} < n^{N/2}$$

или

$$\log n + N + 2n + 1 < (N/2) \log n.$$

Видно, что можно положить $N = \lceil cn / \log n \rceil$ при достаточно большом c, и условие леммы будет выполнено при всех n.

11. Теорема о согласованных кодах

Теорема 2. *Пусть x, y, z —произвольные слова длины менее n. Тогда существуют слова p и q с такими свойствами:*

$$|p| = K(x|y);\ |q| = K(x|z);$$
более короткое из слов p и q является началом более длинного;
$$K(p|x) = O(\log n);\ K(q|x) = O(\log n);$$
$$K(x|p, y) = O(\log n);\ K(x|q, z) = O(\log n).$$

(Константы в $O(\log n)$-обозначениях не зависят от n, x, y, z.)

Доказательство теоремы.

Построим N отображений $\chi_1, \ldots, \chi_N : X \to \mathbb{B}^n$ с указанными в лемме 2 свойствами, где X — множество всех слов длины менее n, в качестве ε взято число $1/n$, а $N = O(n/\log n)$.

Взяв набор хеш-функций первым, который найдётся при поиске в каком-либо естественном порядке, мы можем предполагать, что его сложность есть $O(\log n)$, поскольку для его задания достаточно указать число n.

Пусть $K(x|y) = a$ и $K(x|z) = b$. Оставляя от хеш-значений только a или b первых битов, мы получим N отображений X в $\mathbb{B}^a$ и в $\mathbb{B}^b$. Эти семейства задают двудольные графы на $X \times \mathbb{B}^a$ и $X \times \mathbb{B}^b$, в которых степень каждой вершины из X равна N (считая кратные рёбра). Нас интересуют ограничения этих графов на $X_y \times \mathbb{B}^a$ и $X_z \times \mathbb{B}^b$, где X_y состоит из слов длины менее n, имеющих сложность не более a относительно y, а X_z состоит из слов длины менее n, имеющих сложность не более b относительно z. Мы выделяем в $\mathbb{B}^a$ хорошие вершины, имеющие не более n^c соседей в X_y; аналогичным образом хорошими вершинами в $\mathbb{B}^b$ мы считаем те, у которых не более n^c соседей в X_z. (Точное значение достаточно большой константы c мы выберем позже.)

Число плохих вершин в обоих случаях не превосходит соответственно

$$2N \cdot 2^a/n^c \text{ и } 2N \cdot 2^b/n^c,$$

так как степень плохой вершины больше n^c, общее число рёбер в графе не больше $|X_y| \cdot N$ (соответственно $|X_z| \cdot N$), а $|X_y| < 2 \cdot 2^a$ и $|X_z| < 2 \cdot 2^b$ (число программ длины не больше m меньше $2 \cdot 2^m$).

Назовём опасными в X_y те вершины, у которых половина или более соседей в графе на $X_y \times \mathbb{B}^a$ являются плохими. Лемма гарантирует, что число опасных вершин меньше

$$2N \cdot 2^a/n^c$$

(мы считаем, что c достаточно велико, поэтому оценка на число плохих вершин меньше $\varepsilon 2^a$, где $\varepsilon = 1/n$). Поскольку опасные вершины можно перечислять, зная n, a и y, сложность любой из них относительно y не превосходит

$$\log(2N \cdot 2^a/n^c) + O(\log n) \leq a - c\log n + O(\log n)$$

(напомним, что $N = O(n/\log n)$). При достаточно большом c все опасные вершины имеют сложность (относительно y) меньше a, и исходное слово x не попадает в их число. Это значит, что больше половины хеш-функций в применении к x порождают хороших соседей в $\mathbb{B}^a$.

Повторяя те же рассуждения для графа в $X_z \times \mathbb{B}^b$, мы получаем, что больше половины хеш-функций дают хороших соседей в $\mathbb{B}^b$. Следовательно, найдётся хеш-функция χ_i, которая даёт хороших соседей в обоих случаях.

Обозначим $[\chi_i(x)]_a$ через p, а $[\chi_i(x)]_b$ через q. Слова p и q согласованы (более короткое является началом более длинного). Кроме того, $K(x|p, y) = O(\log n)$, так как при известных y и p для перечисления элементов X_y, являющихся соседями p, достаточно знать $O(\log n)$ битов (число n; заметим, что a восстанавливается по p), а для задания x надо ещё указать порядковый номер x в этом перечислении, который не превосходит n^c и потому требует для своего задания также не более $O(\log n)$ битов. Аналогичным образом $K(x|q, z) = O(\log n)$, что завершает доказательство теоремы 2.

12. Сколько требуется хеш-функций?

Следующая теорема показывает, что количество хеш-функций, построенных в теореме 1, а тем более в теореме 2, не может быть значительно уменьшено.

Теорема 3. *Для любого положительного числа c, любого слова x и любого множества P мощности меньше $K(x)/c\log K(x)$, состоящего из слов длины $\lceil K(x)/2\rceil$, найдётся слово y, для которого*

$$\textbf{(i)}\quad K(y) < O(K(x));$$
$$\textbf{(ii)}\quad K(x|y) < K(x)/2;$$
$$\textbf{(iii)}(\forall p \in P)K(x|y,p) > (c - O(1))\log K(x).$$

Доказательство. Пусть $P = \{p_1, \ldots, p_j\}$ и $j < K(x)/c\log K(x)$. Обозначим через v_i начальный отрезок p_i, имеющий длину $\lfloor c\log K(x)/3\rfloor$. Пусть w — конкатенация слов $v_1, \ldots, v_j$. Тогда длина слова w меньше $j\lfloor c\log K(x)/3\rfloor < K(x)/3$. Рассмотрим значение величины $K(x|w,z)$ для слов z, пробегающих начала слова x. Когда длина z меняется на 1, значение $K(x|w,z)$ меняется не более чем на константу. С одной стороны,

$$K(x|w,\Lambda) > K(x) - K(w) - O(\log K(x)) > K(x) - |w| - O(\log K(x)).$$

Учитывая оценку на длину w, при достаточно больших $K(x)$ получаем $K(x|w,\Lambda) > K(x)/2$. С другой стороны, $K(x|w,x) < O(1)$. Поэтому среди начал слова x существует такое слово z_0, что $K(x)/2 > K(x|w,z_0) > K(x)/2 - O(1)$. Возьмём в качестве y пару $\langle w, z_0\rangle$ (точнее, её код). Проверим (i):

$$K(y) < K(w) + K(z_0) + O(\log K(x)) < K(x)/3 + K(x) + O(\log K(x)).$$

Проверим (ii):

$$K(x|y) = K(x|w,z_0) < K(x)/2.$$

Проверим (iii). Для каждого i слово y содержит «много информации» о p_i. А именно,

$$K(p_i|y) < K(x)/2 - c\log K(x)/3 + O(\log K(x)).$$

(При известном w для получения p_i достаточно указать начало и конец вхождения слова v_i в слово w и остаток слова p_i после отбрасывания v_i.) Используя последнее неравенство, выводим для каждого i

$$K(x|y,p_i) > K(x|y) - K(p_i|y) - O(\log K(x)) > (K(x)/2 - O(1)) - (K(x)/2 - c\log K(x)/3 + O(\log K(x))) - O(\log K(x)) = (c$$

Теорема доказана.

13. Заключительные замечания

1. Формулировка и доказательство теоремы 2 легко обобщаются на три условия вместо двух (а также на большее, но полиномиально зависящее от n число условий).

2. В теоремах 1 и 2 строится код слова x при известном слове y (мом слове z). В условиях этих теорем предполагалось, что длины слов x, y, z меньше n. Однако в приведённых доказательствах ограничения на y и z никак не используются. Что касается x, то ограничение $|x|nN$ можно заменить на $K(x) < n$.

Доказательства приведённых выше результатов были рассказаны в сентябре 1999 года на Колмогоровском семинаре (в Москве). Предварительная краткая версия этой статьи опубликована в [5]. Автор благодарен Александру Шеню, который любезно согласился помочь сделать изложение более понятным; настоящая статья в основном соответствует его тексту.

Литература

[1] И. Чисар, Я. Кёрнер, *Теория информации. Теоремы кодирования для дискретных систем без памяти.* Перевод с английского С. И. Гельфанда и Л. Е. Филипповой под редакцией Р. Л. Добрушина, М.:Мир, 1985. 397 с.

[2] C. H. Bennett, P. Gács, M. Li, P. M. B. Vitányi, W. H. Zurek. Information Distance. *IEEE Transactions on Information Theory*, **44** (1998), No. 3, с. 1407–1423.

[3] L. Fortnow, S. Laplante, Nearly optimal language compression using extractors, *15th Annual Symposium in Theoretical Aspects of Computer Science, Paris, France, 25–27 February 1998*, volume 1373 of *Lecture Notes in Computer Science*, p. 84–93, Springer-Verlag, 1998.

[4] K.Yu. Gorbunov, On a complexity of the formula $((A \vee B) \to C)$, *Theoretical Computer Science*, **207** (1988), p. 383–386.

[5] Andrej Muchnik, Alexej Semenov, Multi-conditional Descriptions and Codes in Kolmogorov Complexity, *Electronic Colloquium on Computational Complexity*, Report No. 15 (2000), January 27, 2000

[6] M. Li, P. Vitányi, *An introduction to Kolmogorov Complexity and Its Applications*, Second edition, Springer-Verlag, 1997.

[7] N. K. Vereshchagin, M. V. Vyugin, Independent minimum length programs to translate between given strings, *Theoretical Computer Science*, this issue.