Докажем, что энтропия пары величин не больше суммы энтропий с использованием префиксных кодов.

Энтропия N-кратного повторения пары величин не может превышать N-кратную сумму энтропий просто в силу наличия префиксного кода для N пар, который сначала описывает все первые компоненты, а потом все вторые компоненты. Разделять их нам не надо, так как это префиксные коды; получившийся код, очевидно, тоже префиксный; ошибка не превышает одного бита и делится на N.

Ожидание префиксной сложности набора из n независимых повторов одного и того же испытания равно $n \times H(\xi) + C$. Действительно, энтропийный префиксный код является одним из способов вычислимо расшифровываемого префиксно корректного описания. с другой стороны, любой способ префиксно корректного описания позволяет построить префиксный код просто выбором первого в словарном порядке из кратчайших описаний для каждого слова. Ранее мы рассматривали вероятностные пространства с заданным набором непересекающихся элементарных событий. Теперь мы рассмотрим свойства бесконечных последовательностей нулей и единиц; мы будем хотеть, чтобы про каждый начальный отрезок можно было сказать, что он порождён независимыми бросаниями симметричной монеты.

Естественными событиями для такого описания будут события вида ``последовательность начинается с $x_1 \ldots x_n$'' с вероятностью по 2^{-n}. К сожалению, просто назвать все такие события элементарными не получится, так как они пересекаются. Назовём их пока что базовыми. С другой стороны, если мы получаем такие события по одному, несложно возвращать в ответ на каждое событие некоторый набор событий так, что никакие два из возвращённых событий не будут пересекаться, а объединение полученных и возвращённых всегда будет такое же, как у полученных событий.

Разумеется, из конечного набора случайных событий можно было бы просто выкинуть все события, вложенные в большие, но для работы с бесконечными (но просто описываемыми) последовательностями событий удобнее исправлять пересечения в порядке поступления. Кроме того, заметим, что любая замена базового события на некоторое разбиение этого базового события на непересекающиеся вложенные в него базовые события даст ту же вероятность.

Вероятностью множества, заданного как объединение непересекающихся базовых множеств назовём сумму ряда вероятностей этих базовых множеств.

Бесконечную последовательность X назовём неслучайной, если существует программа, которая по любому ε печатает (возможно, бесконечное) количество базовых событий, сумма вероятностей которых не превосходит ε, и объединение которых содержит X. Остальные последовательности называются случайными.

Другое определение: последовательность называется случайной, если существует такое программно порождаемое перечисление событий суммарной вероятности 1, что данная последовательность лежит в бесконечном их количестве.

Эти два определения эквивалентны. Действительно, если мы умеем породить список базовых событий с суммой вероятностей C, который покрывает нашу последовательность бесконечно много раз, то наша последовательность в какой-то момент будет покрыта хотя бы C/ε раз --- но суммарная мера C не позволяет базовым событиям больше ε быть покрытым так много раз. В обратную сторону: пусть мы для каждого ε умеем выписывать последовательность базовых событий такой суммарной длины, хоть раз покрывающих нашу последовательность. Тогда для любого δ мы можем параллельно запустить процесс выписывания таких последовательностей базовых событий для $\varepsilon = \frac{\delta}{2}, \frac{\delta}{4}, \ldots$.

На самом деле, как и в случае с префиксной сложностью, существует одна конкретная программа, порождающая

бесконечное количество событий суммарной вероятностью не больше 1 и покрывающая каждую неслучайную последовательность бесконечно много раз. Для этого достаточно перебрать все программы, давая каждой следующей вдвое меньший параметр и не позволяя выводить базовые множества сверх вероятности, заданной этим параметром. Среди программ встретятся все программы, умеющие бесконечно много раз покрыть неслучайную последовательность бесконечно много раз при сколь угодно малой суммарной вероятности.

Можно показать и связь между сложностью описания начал последовательности и случайностью последовательности. Правда, для этого нужно другое определение сложности --- (обычная) колмогоровская сложность. В отличие от префиксной сложности, при определении колмогоровской сложности программа получает на вход просто слово известной длины; на расшифровку слова и его продолжений никаких правил не накладывается.

Сравним это определение с префиксной сложностью. С одной стороны, в отличие от префиксной сложности, при описании пары мы не можем просто приписывать описания, так как границу между ними, нельзя найти. Поэтому для описания пары слов придётся потратить ещё логарифмическое количество битов на запись длины описания первого из них (причём так, чтобы граница записи длины определялась однозначно). С другой стороны, при описании одиночного слова у нас есть естественная оценка сложности ``длина слова плюс константа": так как программа знает длину слова, то она может просто переписать его в ответ.

Заметим, что так как описаний короче, чем $n-1$ не более 2^n-1 штук, большинство слов длины n имеют кратчайшие описания не короче $n-2$.

Мы видим, что колмогоровская сложность большинства слов отличается от n не больше, чем на некоторую константу. Для неслучайных слов верно обратное.

Действительно, так как у нас есть программа, перечисляющая базовые события с суммарной вероятностью 1 и покрывающая каждую последовательность бесконечно много раз, начал длины 2^n, покрытых 2^k всего лишь $2^{2^n-2^k}$ штук; каждое описывается указанием n, k и номера среди них, что требует $C+2\log n+2\log k+2^n-2^k$ битов и заметно меньше 2^n битов. С другой стороны, для любой неслучайной последовательности и любого k достаточно большое n найдётся.

Так же можно показать наличие коротких описаний у случайных последовательностей с точки зрения префиксной сложности. Но для того, чтобы дать определение случайности в точности совпадающее с ранее данным, пришлось бы определять монотонную сложность, которая устроена ещё чуть сложнее префиксной. Но на практике не так важно, как определять случайность: если мы видим, что миллион бросаний монетки сжался архиватором на 10%, это столь маловероятно для настоящих случайных последовательностей, что в видимой части вселенной физика не сможет описать нужное количество экспериментов (планковский объём и планковское время очень малы, а возраст и размер вселенной велики --- но не настолько).

Докажем теперь, что способ определения префиксной сложности не важен (напомним, что один способ запрещает программе распаковки давать ответ на продолжениях корректных описаний, а второй способ требует от неё давать на продолжениях корректного описания тот же ответ).

Для этого определим ещё одно понятие --- *априорную вероятность*.

Пусть у нас есть программа с доступом к источнику независимых случайных битов. Она может никогда не остановиться или напечатать какое-то конечное слово. Рассмотрим распределение вероятностей для слов, которые она может напечатать.

Можно рассмотреть и программы, которые печатают согласованные в совокупности утверждения вида ``слово x имеет вероятность не меньше p'' и рассматривать предельные вероятности слов (остаток от единицы даст вероятность зависания программы).

Это одно и то же, так как можно просто моделировать работу программы на всех возможных последовательностях (разветвляясь при считывании случайного бита) и выписывать оценки на вероятность --- или же раскладывать оценки на вероятность по ветвям дерева декодирования.

Теорема: обе префиксных сложности равны логарифму априорной вероятности.

Беспрефиксную функцию (продолжение корректного описания не может быть корректным) можно просто применять ко всем началам аргумента для получения префиксно корректной (продолжение описания описывает то же самое).

По префиксно корректной функции можно построить машину, порождающую по случайной последовательности битов конечное слово, пробуя расшифровать все начала параллельно.

Пусть теперь мы узнаём оценки снизу на вероятность слова, и должны выделить беспрефиксные коды. Мы имеем право уменьшить все оценки в константу раз. Округлим запросы вниз до степеней двойки. Поделим их пополам, чтобы мы могли с каждым ε выделять и $\frac{\varepsilon}{2}, \frac{\varepsilon}{4}, \ldots$. Представим имеющееся у нас место на двоичном дереве в виде блоков попарно разного размера. Каждое слово будем кодировать, откусывая кусок от самого маленького подходящего блока. Куски, на которые развалится испорченный кусок имеют размеры от исходного до размера запроса, которые не встречались до этого. С другой стороны, каждый запрос выполним, так как сумма всех отрицательных степеней двойки, меньших данной, всё равно меньше её.