Kolmogorov complexity as a language

Alexander Shen
LIF CNRS, Marseille; on leave from
ИППИ РАН, Москва

Kolmogorov complexity

Kolmogorov complexity

- A powerful tool

Kolmogorov complexity

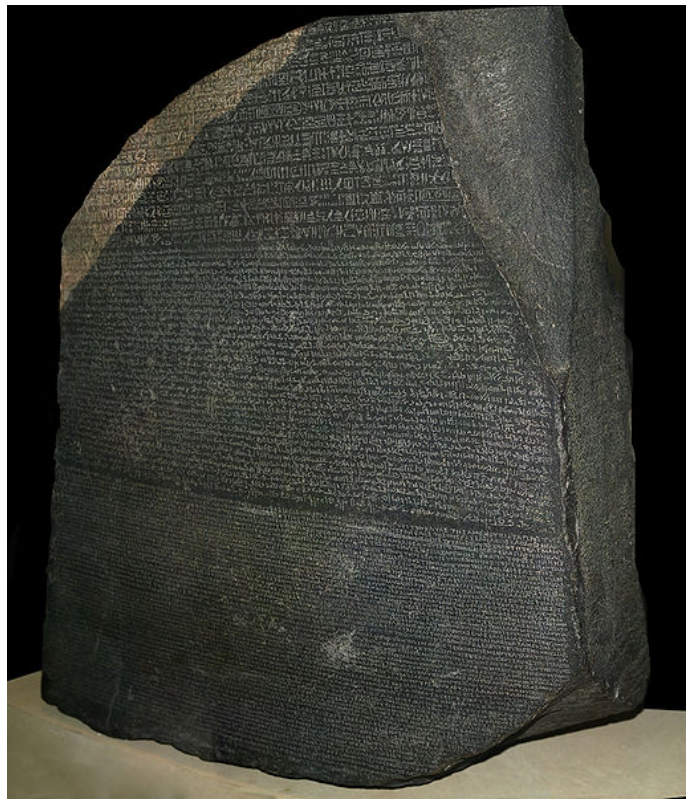
- A powerful tool
- Just a way to reformulate arguments

Kolmogorov complexity

- A powerful tool
- Just a way to reformulate arguments
- three languages: combinatorial/algorithmic/probabilistic

Kolmogorov complexity

- A powerful tool
- Just a way to reformulate arguments
- three languages: combinatorial/algorithmic/probabilistic



Reminder and notation

Reminder and notation

- $K(x) =$ minimal length of a program that produces x

Reminder and notation

- $K(x) =$ minimal length of a program that produces x
- $K_D(x) = \min\{|p| : D(p) = x\}$

Reminder and notation

- $K(x) =$ minimal length of a program that produces x
- $K_D(x) = \min\{|p| : D(p) = x\}$
- depends on the interpreter D

Reminder and notation

- $K(x) =$ minimal length of a program that produces x
- $K_D(x) = \min\{|p| : D(p) = x\}$
- depends on the interpreter D
- optimal D makes it minimal up to $O(1)$ additive term

Reminder and notation

- $K(x) =$ minimal length of a program that produces x
- $K_D(x) = \min\{|p| : D(p) = x\}$
- depends on the interpreter D
- optimal D makes it minimal up to $O(1)$ additive term
- Variations:

Reminder and notation

- $K(x) =$ minimal length of a program that produces x
- $K_D(x) = \min\{|p| : D(p) = x\}$
- depends on the interpreter D
- optimal D makes it minimal up to $O(1)$ additive term
- Variations:

	p = string	p = prefix of a sequence
x = string	plain $K(x)$, $C(x)$	prefix $KP(x)$, $K(x)$
x = prefix of a sequence	decision $KR(x)$, $KD(x)$	monotone $KM(x)$, $Km(x)$

Reminder and notation

- $K(x) =$ minimal length of a program that produces x
- $K_D(x) = \min\{|p| : D(p) = x\}$
- depends on the interpreter D
- optimal D makes it minimal up to $O(1)$ additive term
- Variations:

	p = string	p = prefix of a sequence
x = string	plain $K(x)$, $C(x)$	prefix $KP(x)$, $K(x)$
x = prefix of a sequence	decision $KR(x)$, $KD(x)$	monotone $KM(x)$, $Km(x)$

Conditional complexity $C(x|y)$:
minimal length of a program $p : y \mapsto x$.

Reminder and notation

- $K(x) =$ minimal length of a program that produces x
- $K_D(x) = \min\{|p| : D(p) = x\}$
- depends on the interpreter D
- optimal D makes it minimal up to $O(1)$ additive term
- Variations:

	p = string	p = prefix of a sequence
x = string	plain $K(x)$, $C(x)$	prefix $KP(x)$, $K(x)$
x = prefix of a sequence	decision $KR(x)$, $KD(x)$	monotone $KM(x)$, $Km(x)$

Conditional complexity $C(x|y)$:
minimal length of a program $p : y \mapsto x$.
There is also a priori probability (in two versions: discrete, on strings; continuous, on prefixes)

Foundations of probability theory

Foundations of probability theory

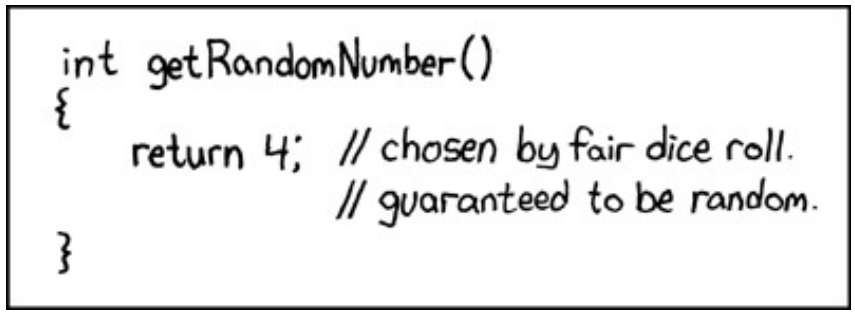
- Random object or random process?

Foundations of probability theory

- Random object or random process?
- “well shuffled deck of cards”: any meaning?

Foundations of probability theory

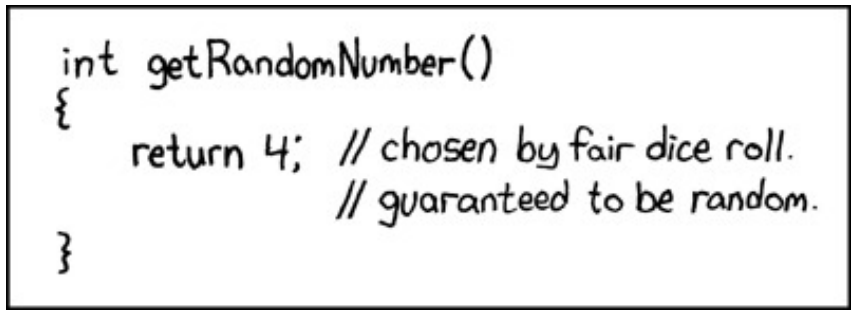
- Random object or random process?
- "well shuffled deck of cards": any meaning?



[xkcd cartoon]

Foundations of probability theory

- Random object or random process?
- “well shuffled deck of cards”: any meaning?

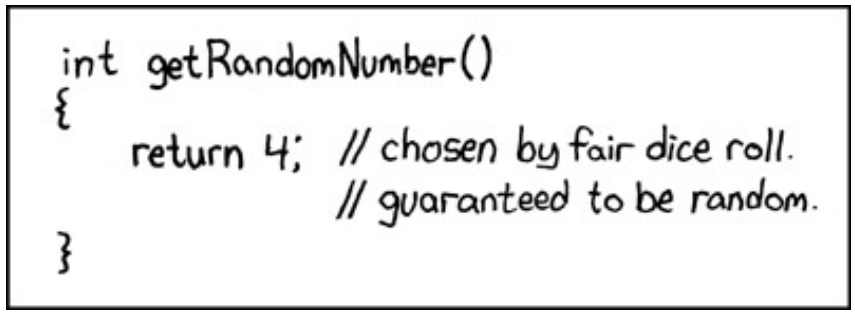


[xkcd cartoon]

- randomness = incompressibility (maximal complexity)

Foundations of probability theory

- Random object or random process?
- "well shuffled deck of cards": any meaning?

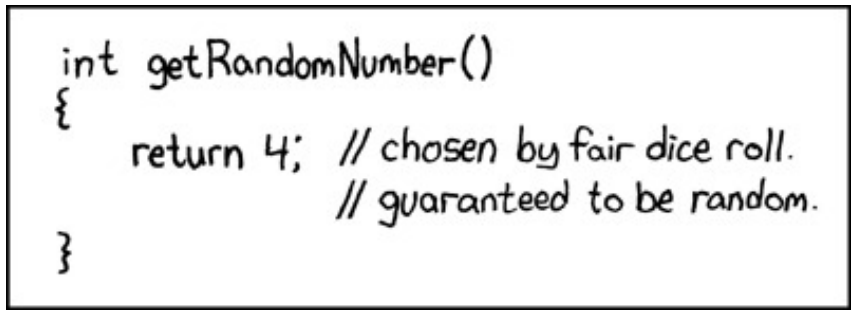


[xkcd cartoon]

- randomness = incompressibility (maximal complexity)
- $\omega = \omega_1\omega_2\ldots$ is random iff $KM(\omega_1\ldots\omega_n) \geq n - O(1)$

Foundations of probability theory

- Random object or random process?
- "well shuffled deck of cards": any meaning?

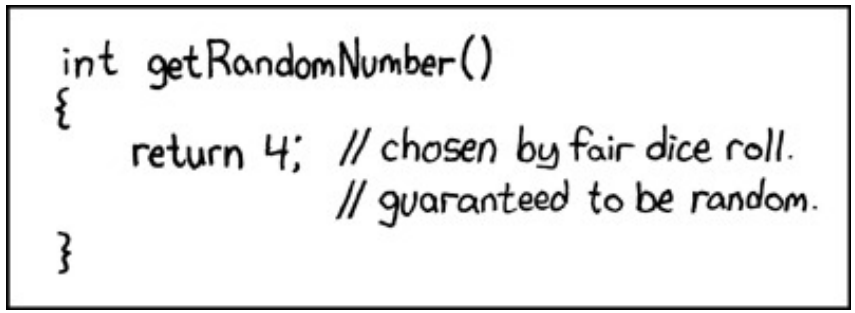


[xkcd cartoon]

- randomness = incompressibility (maximal complexity)
- $\omega = \omega_1\omega_2\ldots$ is random iff $KM(\omega_1\ldots\omega_n) \geq n - O(1)$
- Classical probability theory: random sequence satisfies the Strong Law of Large Numbers with probability 1

Foundations of probability theory

- Random object or random process?
- "well shuffled deck of cards": any meaning?

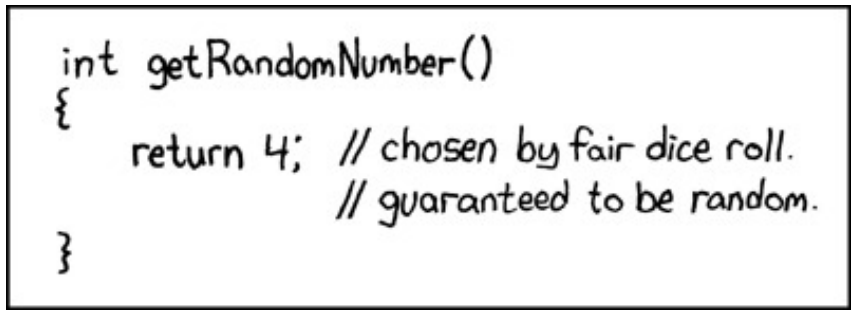


[xkcd cartoon]

- randomness = incompressibility (maximal complexity)
- $\omega = \omega_1\omega_2\ldots$ is random iff $KM(\omega_1\ldots\omega_n) \geq n - O(1)$
- Classical probability theory: random sequence satisfies the Strong Law of Large Numbers with probability 1
- Algorithmic version: every (algorithmically) random sequence satisfies SLLN

Foundations of probability theory

- Random object or random process?
- "well shuffled deck of cards": any meaning?



[xkcd cartoon]

- randomness = incompressibility (maximal complexity)
- $\omega = \omega_1\omega_2\ldots$ is random iff $KM(\omega_1\ldots\omega_n) \geq n - O(1)$
- Classical probability theory: random sequence satisfies the Strong Law of Large Numbers with probability 1
- Algorithmic version: every (algorithmically) random sequence satisfies SLLN
- algorithmic $\Rightarrow$ classical: Martin-Löf random sequences form a set of measure 1.

Sampling random strings (S.Aaronson)

Sampling random strings (S.Aaronson)

- A device that (being switched on) produces N-bit string and stops

Sampling random strings (S.Aaronson)

- A device that (being switched on) produces N-bit string and stops
- “The device produces a random string”: what does it mean?

Sampling random strings (S.Aaronson)

- A device that (being switched on) produces N-bit string and stops
- “The device produces a random string”: what does it mean?
- classical: the output distribution is close to the uniform one

Sampling random strings (S.Aaronson)

- A device that (being switched on) produces N-bit string and stops
- “The device produces a random string”: what does it mean?
- classical: the output distribution is close to the uniform one
- effective: with high probability the output string is incompressible

Sampling random strings (S.Aaronson)

- A device that (being switched on) produces N-bit string and stops
- “The device produces a random string”: what does it mean?
- classical: the output distribution is close to the uniform one
- effective: with high probability the output string is incompressible
- not equivalent if no assumptions about the device

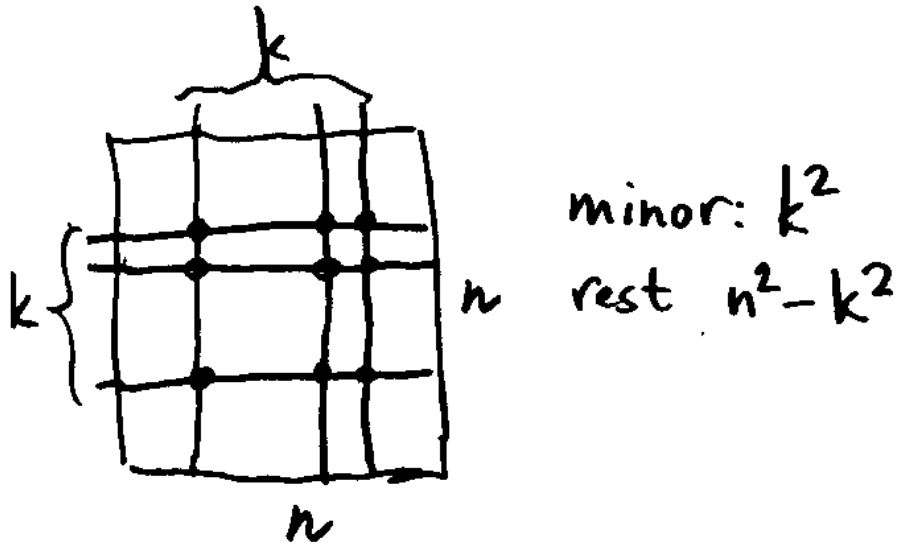
Sampling random strings (S.Aaronson)

- A device that (being switched on) produces N-bit string and stops
- “The device produces a random string”: what does it mean?
- classical: the output distribution is close to the uniform one
- effective: with high probability the output string is incompressible
- not equivalent if no assumptions about the device
- but are related under some assumptions

Example: matrices without uniform minors

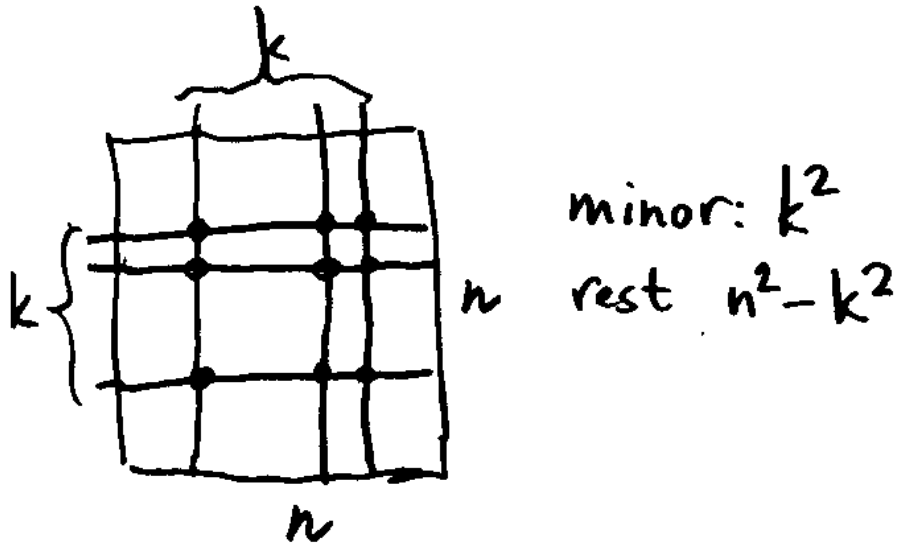
Example: matrices without uniform minors

- $k \times k$ minor of $n \times n$ Boolean matrix: select k rows and k columns



Example: matrices without uniform minors

- $k \times k$ minor of $n \times n$ Boolean matrix: select k rows and k columns



- minor is uniform if it is all-0 or all-1.
- claim: there is a $n \times n$ bit matrix without $k \times k$ uniform minors for $k = 3 \log n$.

Counting argument and complexity reformulation

Counting argument and complexity reformulation

- $\leq n^k \times n^k$ positions of the minor [$k = 3 \log n$]

Counting argument and complexity reformulation

- $\leq n^k \times n^k$ positions of the minor [$k = 3 \log n$]
- 2 types of uniform minors (0/1)

Counting argument and complexity reformulation

- $\leq n^k \times n^k$ positions of the minor [$k = 3 \log n$]
- 2 types of uniform minors (0/1)
- $2^{n^2-k^2}$ possibilities for the rest

Counting argument and complexity reformulation

- $\leq n^k \times n^k$ positions of the minor [$k = 3 \log n$]
- 2 types of uniform minors (0/1)
- $2^{n^2-k^2}$ possibilities for the rest
- $n^{2k} \times 2 \times 2^{n^2-k^2}$ =

Counting argument and complexity reformulation

- $\leq n^k \times n^k$ positions of the minor [$k = 3 \log n$]
- 2 types of uniform minors (0/1)
- $2^{n^2-k^2}$ possibilities for the rest
- $n^{2k} \times 2 \times 2^{n^2-k^2} = 2^{\log n \times 2 \times 3 \log n + 1 + (n^2 - 9 \log^2 n)} < 2^{n^2}$.

Counting argument and complexity reformulation

- $\leq n^k \times n^k$ positions of the minor [$k = 3 \log n$]
- 2 types of uniform minors (0/1)
- $2^{n^2-k^2}$ possibilities for the rest
- $n^{2k} \times 2 \times 2^{n^2-k^2} = 2^{\log n \times 2 \times 3 \log n + 1 + (n^2 - 9 \log^2 n)} < 2^{n^2}$.

- $\log n$ bits to specify a column or row: $2k \log n$ bits in total

Counting argument and complexity reformulation

- $\leq n^k \times n^k$ positions of the minor [$k = 3 \log n$]
- 2 types of uniform minors (0/1)
- $2^{n^2-k^2}$ possibilities for the rest
- $n^{2k} \times 2 \times 2^{n^2-k^2} = 2^{\log n \times 2 \times 3 \log n + 1 + (n^2 - 9 \log^2 n)} < 2^{n^2}$.

- $\log n$ bits to specify a column or row: $2k \log n$ bits in total
- one additional bit to specify the type of minor $(0/1)$

Counting argument and complexity reformulation

- $\leq n^k \times n^k$ positions of the minor [$k = 3 \log n$]
- 2 types of uniform minors (0/1)
- $2^{n^2-k^2}$ possibilities for the rest
- $n^{2k} \times 2 \times 2^{n^2-k^2} = 2^{\log n \times 2 \times 3 \log n + 1 + (n^2 - 9 \log^2 n)} < 2^{n^2}$.

- $\log n$ bits to specify a column or row: $2k \log n$ bits in total
- one additional bit to specify the type of minor $(0/1)$
- $n^2 - k^2$ bits to specify the rest of the matrix

Counting argument and complexity reformulation

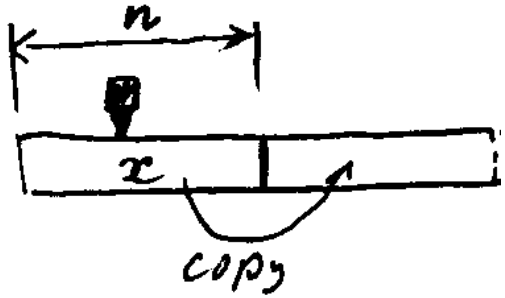
- $\leq n^k \times n^k$ positions of the minor [$k = 3 \log n$]
- 2 types of uniform minors (0/1)
- $2^{n^2-k^2}$ possibilities for the rest
- $n^{2k} \times 2 \times 2^{n^2-k^2} = 2^{\log n \times 2 \times 3 \log n + 1 + (n^2 - 9 \log^2 n)} < 2^{n^2}$.

- $\log n$ bits to specify a column or row: $2k \log n$ bits in total
- one additional bit to specify the type of minor $(0/1)$
- $n^2 - k^2$ bits to specify the rest of the matrix
- $2k \log n + 1 + (n^2 - k^2) = 6 \log^2 n + 1 + (n^2 - 9 \log^2 n) < n^2$.

One-tape Turing machines

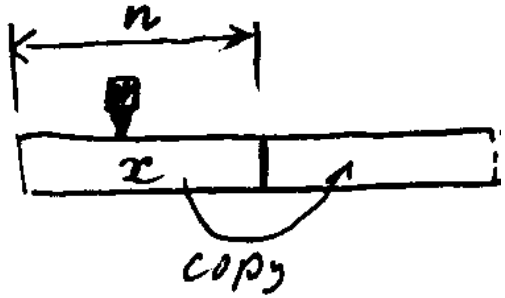
One-tape Turing machines

- copying n-bit string on 1-tape TM requires $\Omega(n^2)$ time

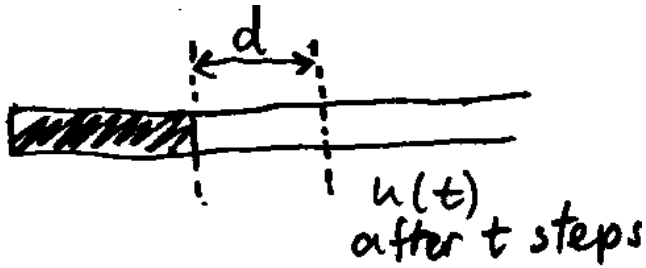


One-tape Turing machines

- copying n-bit string on 1-tape TM requires $\Omega(n^2)$ time



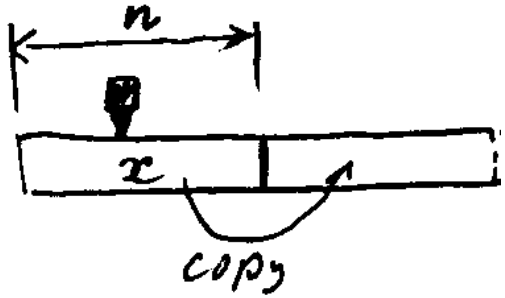
- complexity version: if initially the tape was empty on the right of the border, then after n steps the complexity of a zone that is d cells far from the border is $O(n/d)$.



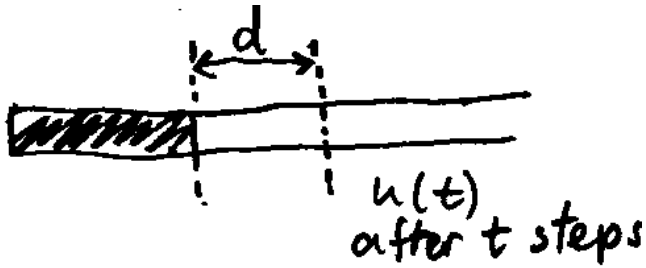
$K(u(t)) \leq O(n/d)$

One-tape Turing machines

- copying n-bit string on 1-tape TM requires $\Omega(n^2)$ time



- complexity version: if initially the tape was empty on the right of the border, then after n steps the complexity of a zone that is d cells far from the border is $O(n/d)$.



$K(u(t)) \leq O(n/d)$

- proof: border guards in each cell of the border security zone write down the contents of the head of TM; each of the records is enough to reconstruct $u(t)$ so the length of it should be $\Omega(K(u(t))$; the sum of lengths does not exceed time

Everywhere complex sequences

Everywhere complex sequences

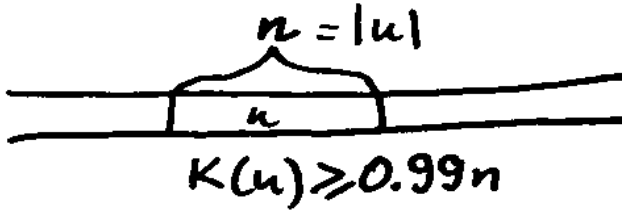
- Random sequence has n-bit prefix of complexity n

Everywhere complex sequences

- Random sequence has n-bit prefix of complexity n
- but some factors (substrings) have small complexity

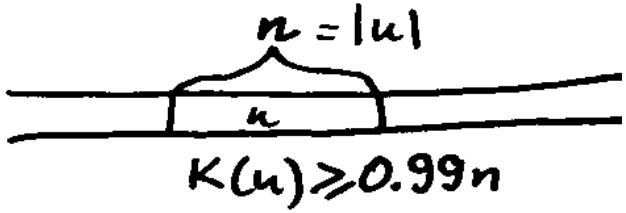
Everywhere complex sequences

- Random sequence has n-bit prefix of complexity n
- but some factors (substrings) have small complexity
- Levin: there exist everywhere complex sequences: every n-bit substring has complexity $0.99n - O(1)$



Everywhere complex sequences

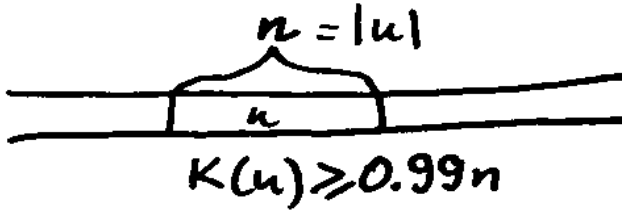
- Random sequence has n-bit prefix of complexity n
- but some factors (substrings) have small complexity
- Levin: there exist everywhere complex sequences: every n-bit substring has complexity $0.99n - O(1)$



- Combinatorial equivalent: Let F be a set of strings that has at most $2^{0.99n}$ strings of length n. Then there is a sequence ω s.t. all sufficiently long substrings of ω are not in F.

Everywhere complex sequences

- Random sequence has n-bit prefix of complexity n
- but some factors (substrings) have small complexity
- Levin: there exist everywhere complex sequences: every n-bit substring has complexity $0.99n - O(1)$

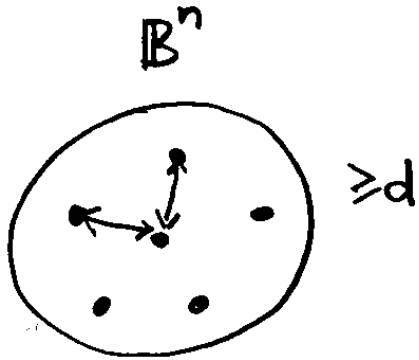


- Combinatorial equivalent: Let F be a set of strings that has at most $2^{0.99n}$ strings of length n. Then there is a sequence ω s.t. all sufficiently long substrings of ω are not in F.
- combinatorial and complexity proofs not just translations of each other (Lovasz lemma, Rumyantsev, Miller, Muchnik)

Gilbert-Varshamov complexity bound

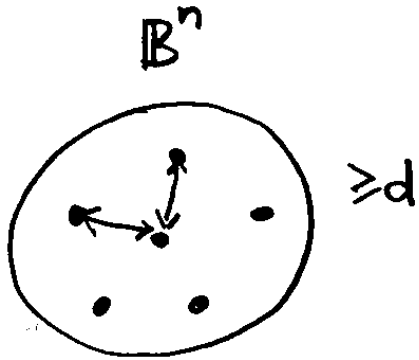
Gilbert-Varshamov complexity bound

- coding theory: how many n-bit strings $x_1, \ldots, x_k$ one can find if Hamming distance between every two is at least d



Gilbert-Varshamov complexity bound

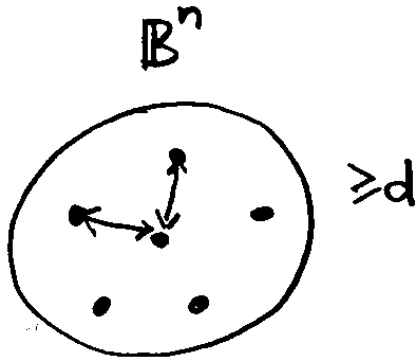
- coding theory: how many n-bit strings $x_1, \ldots, x_k$ one can find if Hamming distance between every two is at least d



- lower bound (Gilbert–Varshamov)

Gilbert-Varshamov complexity bound

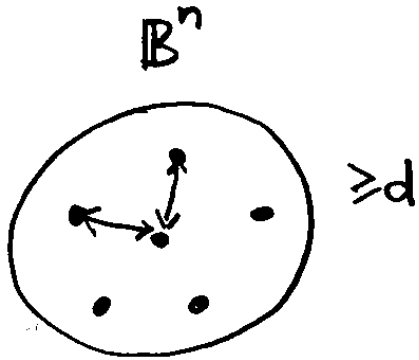
- coding theory: how many n-bit strings $x_1, \ldots, x_k$ one can find if Hamming distance between every two is at least d



- lower bound (Gilbert–Varshamov)
- then $< d/2$ changed bits are harmless

Gilbert-Varshamov complexity bound

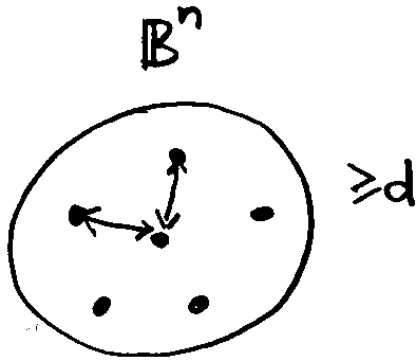
- coding theory: how many n-bit strings $x_1, \ldots, x_k$ one can find if Hamming distance between every two is at least d



- lower bound (Gilbert–Varshamov)
- then $< d/2$ changed bits are harmless
- but bit insertion or deletions could be

Gilbert-Varshamov complexity bound

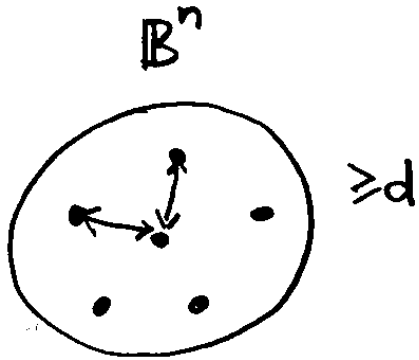
- coding theory: how many n-bit strings $x_1, \ldots, x_k$ one can find if Hamming distance between every two is at least d



- lower bound (Gilbert–Varshamov)
- then $< d/2$ changed bits are harmless
- but bit insertion or deletions could be
- general requirement: $C(x_i|x_j) \geq d$

Gilbert-Varshamov complexity bound

- coding theory: how many n-bit strings $x_1, \ldots, x_k$ one can find if Hamming distance between every two is at least d



- lower bound (Gilbert–Varshamov)
- then $< d/2$ changed bits are harmless
- but bit insertion or deletions could be
- general requirement: $C(x_i|x_j) \geq d$
- generalization of GV bound: d-separated family of size $\Omega(2^{n-d})$

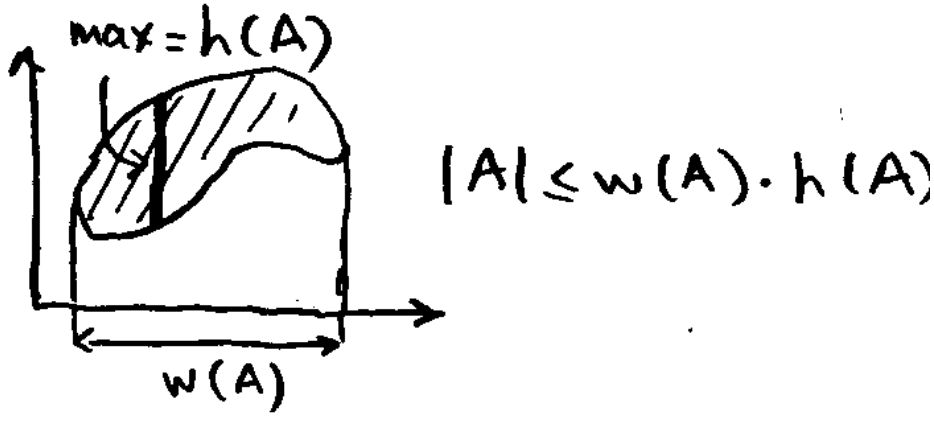
Inequalities for complexities and combinatorial interpretation

Inequalities for complexities and combinatorial interpretation

- $C(x, y) \leq C(x) + C(y|x) + O(\log)$

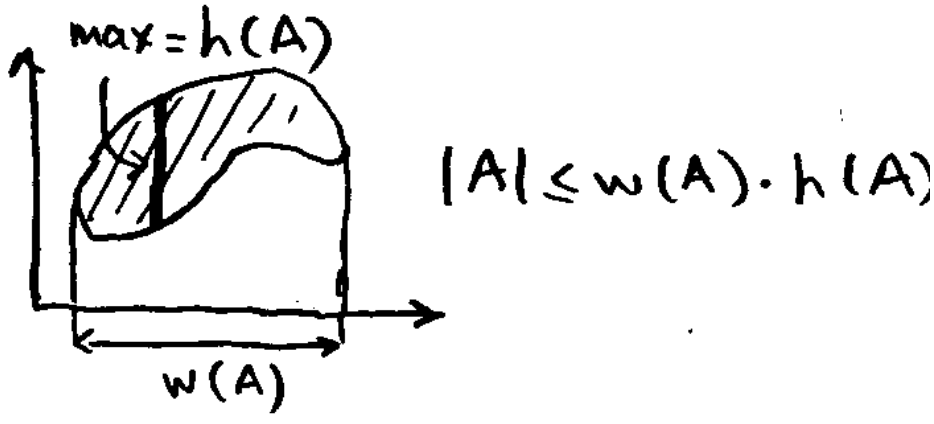
Inequalities for complexities and combinatorial interpretation

- $C(x, y) \leq C(x) + C(y|x) + O(\log)$



Inequalities for complexities and combinatorial interpretation

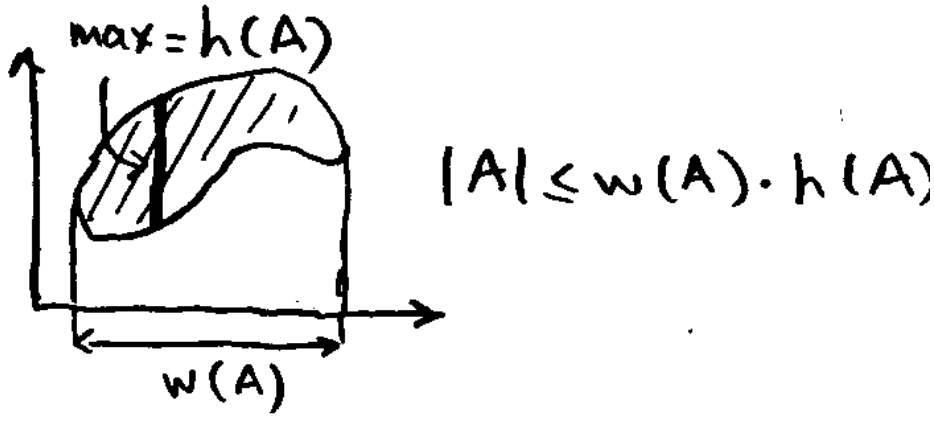
- $C(x, y) \leq C(x) + C(y|x) + O(\log)$



- $C(x, y) \geq C(x) + C(y|x) + O(\log)$

Inequalities for complexities and combinatorial interpretation

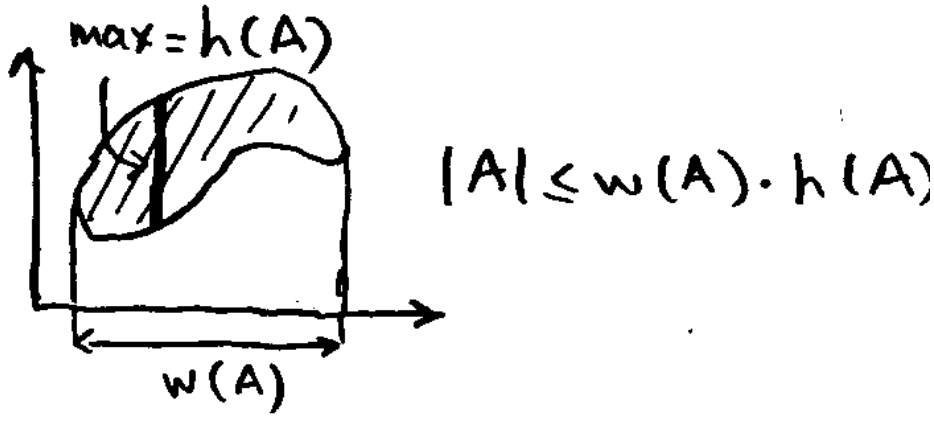
- $C(x, y) \leq C(x) + C(y|x) + O(\log)$



- $C(x, y) \geq C(x) + C(y|x) + O(\log)$
- $C(x, y) < k + l \Rightarrow C(x) < k + O(\log)$ or $C(y|x) < l + O(\log)$

Inequalities for complexities and combinatorial interpretation

- $C(x, y) \leq C(x) + C(y|x) + O(\log)$



- $C(x, y) \geq C(x) + C(y|x) + O(\log)$
- $C(x, y) < k + l \Rightarrow C(x) < k + O(\log)$ or $C(y|x) < l + O(\log)$
- every set A of size $< 2^{k+l}$ can be split into two parts $A = A_1 \cup A_2$ such that $w(A_1) \leq 2^k$ and $h(A_2) \leq 2^l$

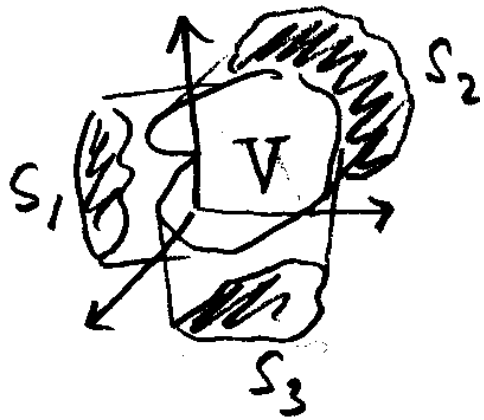
One more inequality

One more inequality

- $2C(x, y, z) \leq C(x, y) + C(y, z) + C(x, z)$

One more inequality

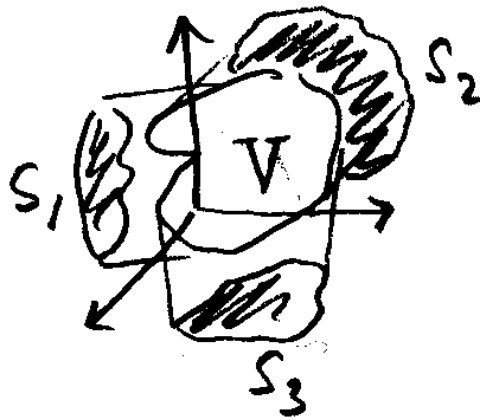
- $2C(x,y,z) \leq C(x,y) + C(y,z) + C(x,z)$



- $V^2 \leq S_1 \times S_2 \times S_3$

One more inequality

- $2C(x,y,z) \leq C(x,y) + C(y,z) + C(x,z)$

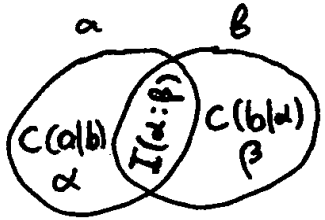


- $V^2 \leq S_1 \times S_2 \times S_3$
- Also for Shannon entropies; special case of Shearer lemma

Common information and graph minors

Common information and graph minors

- mutual information: $I(a : b) = C(a) + C(b) - C(a, b)$

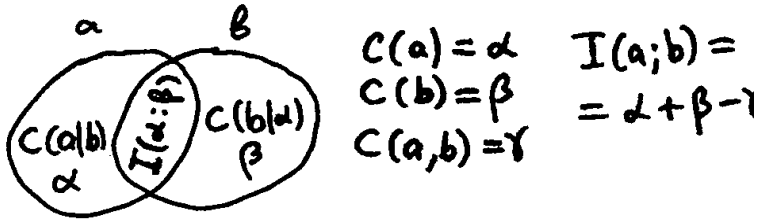


$C(a) = \alpha$
$C(b) = \beta$
$C(a,b) = \gamma$

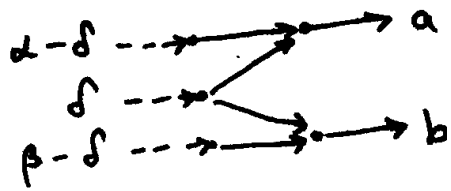
$I(a;b) =$
$= \alpha + \beta - \gamma$

Common information and graph minors

- mutual information: $I(a : b) = C(a) + C(b) - C(a, b)$

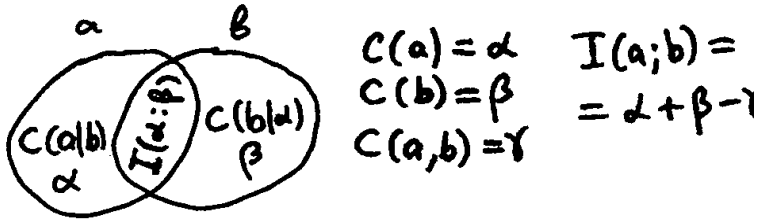


- common information:

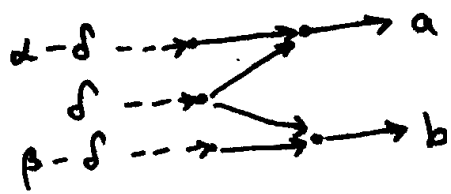


Common information and graph minors

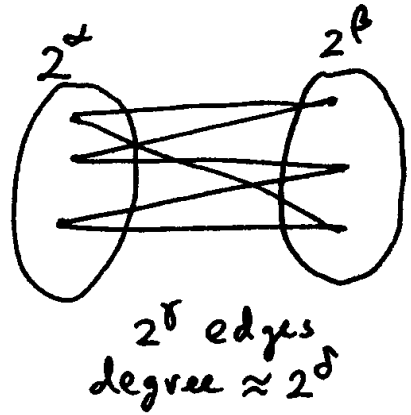
- mutual information: $I(a : b) = C(a) + C(b) - C(a, b)$



- common information:



- combinatorial: graph minors

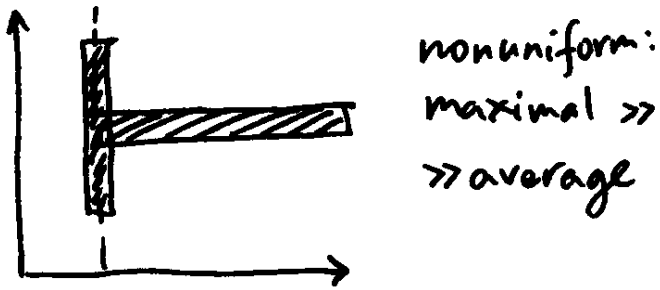


can the graph be covered by 2^δ minors of size $2^{\alpha-\delta} \times 2^{\beta-\delta}$?

Almost uniform sets

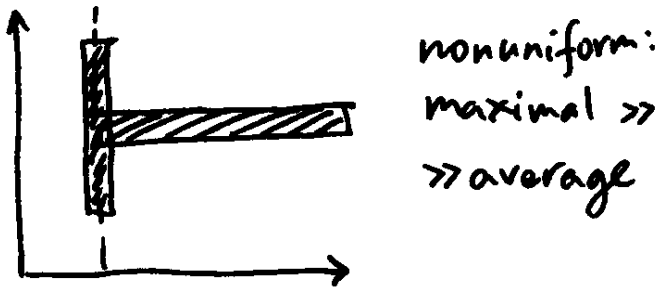
Almost uniform sets

- nonuniformity= (maximal section)/(average section)



Almost uniform sets

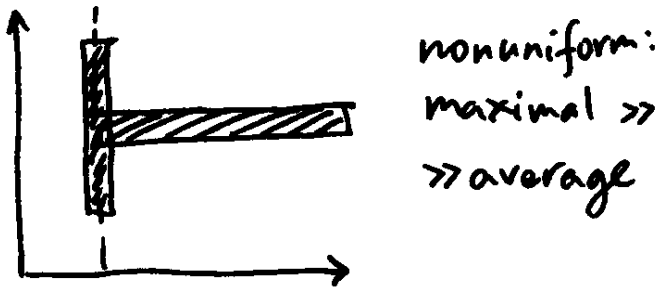
- nonuniformity= (maximal section)/(average section)



- Theorem: every set of N elements can be represented as union of $\mathrm{polylog}(N)$ sets whose nonuniformity is $\mathrm{polylog}(N)$.

Almost uniform sets

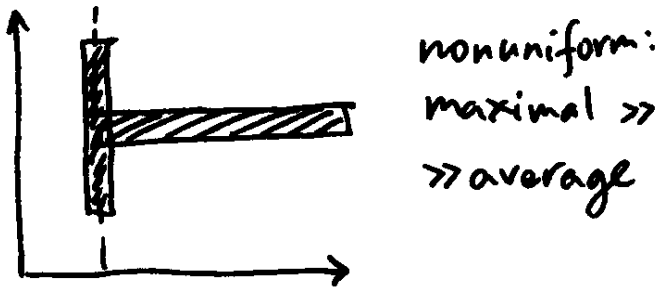
- nonuniformity= (maximal section)/(average section)



- Theorem: every set of N elements can be represented as union of $\mathrm{polylog}(N)$ sets whose nonuniformity is $\mathrm{polylog}(N)$.
- multidimensional version

Almost uniform sets

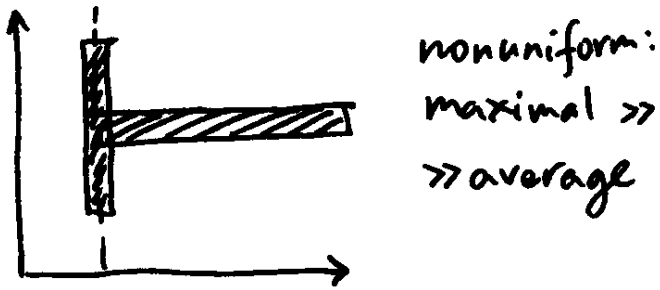
- nonuniformity= (maximal section)/(average section)



- Theorem: every set of N elements can be represented as union of $\mathrm{polylog}(N)$ sets whose nonuniformity is $\mathrm{polylog}(N)$.
- multidimensional version
- how to construct parts using Kolmogorov complexity: take strings with given complexity bounds

Almost uniform sets

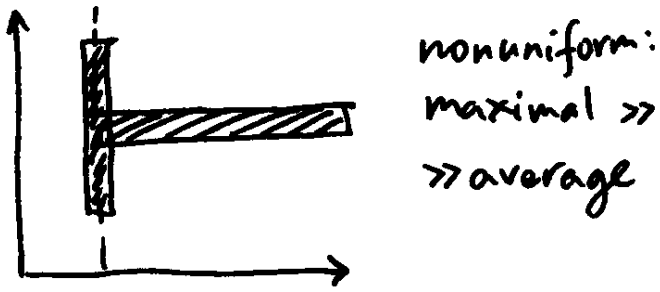
- nonuniformity= (maximal section)/(average section)



- Theorem: every set of N elements can be represented as union of $\mathrm{polylog}(N)$ sets whose nonuniformity is $\mathrm{polylog}(N)$.
- multidimensional version
- how to construct parts using Kolmogorov complexity: take strings with given complexity bounds
- so simple that it is not clear what is the combinatorial translation

Almost uniform sets

- nonuniformity= (maximal section)/(average section)



- Theorem: every set of N elements can be represented as union of $\mathrm{polylog}(N)$ sets whose nonuniformity is $\mathrm{polylog}(N)$.
- multidimensional version
- how to construct parts using Kolmogorov complexity: take strings with given complexity bounds
- so simple that it is not clear what is the combinatorial translation
- but combinatorial argument exists (and gives even a stronger result)

Shannon coding theorem

Shannon coding theorem

- ξ is a random variable; k values, probabilities $p_1, \ldots, p_k$

Shannon coding theorem

- ξ is a random variable; k values, probabilities $p_1, \ldots, p_k$
- ξ^N: N independent trials of ξ

Shannon coding theorem

- ξ is a random variable; k values, probabilities $p_1, \ldots, p_k$
- ξ^N: N independent trials of ξ
- Shannon's informal question: how many bits are needed to encode a "typical" value of ξ^N?

Shannon coding theorem

- ξ is a random variable; k values, probabilities $p_1, \ldots, p_k$
- ξ^N: N independent trials of ξ
- Shannon's informal question: how many bits are needed to encode a "typical" value of ξ^N?
- Shannon's answer: $NH(\xi)$, where

$$H(\xi) = p_1 \log(1/p_1) + \ldots + p_n \log(1/p_n).$$

Shannon coding theorem

- ξ is a random variable; k values, probabilities $p_1, \ldots, p_k$
- ξ^N: N independent trials of ξ
- Shannon's informal question: how many bits are needed to encode a "typical" value of ξ^N?
- Shannon's answer: $NH(\xi)$, where

$$H(\xi) = p_1 \log(1/p_1) + \ldots + p_n \log(1/p_n).$$

- formal statement is a bit complicated

Shannon coding theorem

- ξ is a random variable; k values, probabilities $p_1, \ldots, p_k$
- ξ^N: N independent trials of ξ
- Shannon's informal question: how many bits are needed to encode a "typical" value of ξ^N?
- Shannon's answer: $NH(\xi)$, where

$$H(\xi) = p_1 \log(1/p_1) + \ldots + p_n \log(1/p_n).$$

- formal statement is a bit complicated
- Complexity version: with high probablity the value of ξ^N has complexity close to $NH(\xi)$.

Complexity, entropy and group size

Complexity, entropy and group size

- $2C(x, y, z) \leq C(x, y) + C(y, z) + C(x, z) + O(\log)$

Complexity, entropy and group size

- $2C(x,y,z) \leq C(x,y) + C(y,z) + C(x,z) + O(\log)$
- The same for entropy:
  $2H(\xi,\eta,\tau) \leq H(\xi,\eta) + H(\xi,\tau) + H(\eta,\tau)$

Complexity, entropy and group size

- $2C(x,y,z) \leq C(x,y) + C(y,z) + C(x,z) + O(\log)$
- The same for entropy:
  $2H(\xi,\eta,\tau) \leq H(\xi,\eta) + H(\xi,\tau) + H(\eta,\tau)$
- ...and even for the sizes of subgroups U, V, W of some finite group G: $2\log(|G|/|U \cap V \cap W|) \leq$
  $\log(|G|/|U \cap V|) + \log(|G|/|U \cap W|) + \log(|G|/|V \cap W|)$.

Complexity, entropy and group size

- $2C(x,y,z) \leq C(x,y) + C(y,z) + C(x,z) + O(\log)$
- The same for entropy:
  $2H(\xi,\eta,\tau) \leq H(\xi,\eta) + H(\xi,\tau) + H(\eta,\tau)$
- ...and even for the sizes of subgroups U, V, W of some finite group G: $2\log(|G|/|U \cap V \cap W|) \leq$
  $\log(|G|/|U \cap V|) + \log(|G|/|U \cap W|) + \log(|G|/|V \cap W|)$.
- in all three cases inequalities are the same (Romashchenko, Chan, Yeung)

Complexity, entropy and group size

- $2C(x,y,z) \le C(x,y) + C(y,z) + C(x,z) + O(\log)$
- The same for entropy:
  $2H(\xi,\eta,\tau) \le H(\xi,\eta) + H(\xi,\tau) + H(\eta,\tau)$
- ...and even for the sizes of subgroups U, V, W of some finite group G: $2\log(|G|/|U\cap V\cap W|) \le$
  $\log(|G|/|U\cap V|) + \log(|G|/|U\cap W|) + \log(|G|/|V\cap W|)$.
- in all three cases inequalities are the same (Romashchenko, Chan, Yeung)
- some of them are quite strange:

$$I(a:b) \le$$
$$\le I(a:b|c)+I(a:b|d)+I(c:d)+I(a:b|e)+I(a:e|b)+I(b:e|a)$$

Complexity, entropy and group size

- $2C(x,y,z) \le C(x,y) + C(y,z) + C(x,z) + O(\log)$
- The same for entropy:
  $2H(\xi,\eta,\tau) \le H(\xi,\eta) + H(\xi,\tau) + H(\eta,\tau)$
- ...and even for the sizes of subgroups U, V, W of some finite group G: $2\log(|G|/|U\cap V\cap W|) \le$
  $\log(|G|/|U\cap V|) + \log(|G|/|U\cap W|) + \log(|G|/|V\cap W|)$.
- in all three cases inequalities are the same (Romashchenko, Chan, Yeung)
- some of them are quite strange:

$$I(a:b) \le$$
$$\le I(a:b|c)+I(a:b|d)+I(c:d)+I(a:b|e)+I(a:e|b)+I(b:e|a)$$

- Related to Romashchenko's theorem: if three last terms are zeros, one can extract common information from a, b, e.

Muchnik and Slepian–Wolf

Muchnik and Slepian–Wolf

- a, b: two strings

Muchnik and Slepian–Wolf

- a, b: two strings
- we look for a program p that maps a to b

Muchnik and Slepian–Wolf

- a, b: two strings
- we look for a program p that maps a to b
- by definition $C(p)$ is at least $C(b|a)$ but could be higher

Muchnik and Slepian–Wolf

- a, b: two strings
- we look for a program p that maps a to b
- by definition $C(p)$ is at least $C(b|a)$ but could be higher
- there exist $p\colon a \mapsto b$ that is simple relative to b, e.g., “map everything to b”

Muchnik and Slepian–Wolf

- a, b: two strings
- we look for a program p that maps a to b
- by definition $C(p)$ is at least $C(b|a)$ but could be higher
- there exist $p\colon a \mapsto b$ that is simple relative to b, e.g., "map everything to b"
- Muchnik theorem: it is possible to combine these two conditions: there exists $p\colon a \mapsto b$ such that $C(p) \approx C(b|a)$ and $C(p|b) \approx 0$

Muchnik and Slepian–Wolf

- a, b: two strings
- we look for a program p that maps a to b
- by definition $C(p)$ is at least $C(b|a)$ but could be higher
- there exist $p\colon a \mapsto b$ that is simple relative to b, e.g., "map everything to b"
- Muchnik theorem: it is possible to combine these two conditions: there exists $p\colon a \mapsto b$ such that $C(p) \approx C(b|a)$ and $C(p|b) \approx 0$
- information theory analog: Wolf–Slepian

Muchnik and Slepian–Wolf

- a, b: two strings
- we look for a program p that maps a to b
- by definition $C(p)$ is at least $C(b|a)$ but could be higher
- there exist $p\colon a \mapsto b$ that is simple relative to b, e.g., "map everything to b"
- Muchnik theorem: it is possible to combine these two conditions: there exists $p\colon a \mapsto b$ such that $C(p) \approx C(b|a)$ and $C(p|b) \approx 0$
- information theory analog: Wolf–Slepian
- similar technique was developed by Fortnow and Laplante (randomness extractors)

Muchnik and Slepian–Wolf

- a, b: two strings
- we look for a program p that maps a to b
- by definition $C(p)$ is at least $C(b|a)$ but could be higher
- there exist $p\colon a \mapsto b$ that is simple relative to b, e.g., "map everything to b"
- Muchnik theorem: it is possible to combine these two conditions: there exists $p\colon a \mapsto b$ such that $C(p) \approx C(b|a)$ and $C(p|b) \approx 0$
- information theory analog: Wolf–Slepian
- similar technique was developed by Fortnow and Laplante (randomness extractors)
- (Romashchenko, Musatov): how to use explicit extractors and derandomization to get space-bounded versions

Computability theory: simple sets

Computability theory: simple sets

- Simple set: enumerable set with infinite complement, but no algorithm can generate infinitely many elements from the complement

Computability theory: simple sets

- Simple set: enumerable set with infinite complement, but no algorithm can generate infinitely many elements from the complement
- Construction using Kolmogorov complexity: a simple string x has $C(x) \leq |x|/2$.

Computability theory: simple sets

- Simple set: enumerable set with infinite complement, but no algorithm can generate infinitely many elements from the complement
- Construction using Kolmogorov complexity: a simple string x has $C(x) \leq |x|/2$.
- Most strings are not simple $\Rightarrow$ infinite complement

Computability theory: simple sets

- Simple set: enumerable set with infinite complement, but no algorithm can generate infinitely many elements from the complement
- Construction using Kolmogorov complexity: a simple string x has $C(x) \leq |x|/2$.
- Most strings are not simple $\Rightarrow$ infinite complement
- Let $x_1, x_2, \ldots$ be a computable sequence of different non-simple strings

Computability theory: simple sets

- Simple set: enumerable set with infinite complement, but no algorithm can generate infinitely many elements from the complement
- Construction using Kolmogorov complexity: a simple string x has $C(x) \leq |x|/2$.
- Most strings are not simple $\Rightarrow$ infinite complement
- Let $x_1, x_2, \ldots$ be a computable sequence of different non-simple strings
- May assume wlog that $|x_i| > i$ and therefore $C(x_i) > i/2$

Computability theory: simple sets

- Simple set: enumerable set with infinite complement, but no algorithm can generate infinitely many elements from the complement
- Construction using Kolmogorov complexity: a simple string x has $C(x) \leq |x|/2$.
- Most strings are not simple $\Rightarrow$ infinite complement
- Let $x_1, x_2, \ldots$ be a computable sequence of different non-simple strings
- May assume wlog that $|x_i| > i$ and therefore $C(x_i) > i/2$
- but to specify x_i we need $O(\log i)$ bits only

Computability theory: simple sets

- Simple set: enumerable set with infinite complement, but no algorithm can generate infinitely many elements from the complement
- Construction using Kolmogorov complexity: a simple string x has $C(x) \leq |x|/2$.
- Most strings are not simple $\Rightarrow$ infinite complement
- Let $x_1, x_2, \ldots$ be a computable sequence of different non-simple strings
- May assume wlog that $|x_i| > i$ and therefore $C(x_i) > i/2$
- but to specify x_i we need $O(\log i)$ bits only
- "Minimal integer that cannot be described in ten English words" (Berry)

Lower semicomputable random reals

Lower semicomputable random reals

- $\sum a_i$ computable converging series with rational terms

Lower semicomputable random reals

- $\sum a_i$ computable converging series with rational terms
- is $\alpha = \sum a_i$ computable ($\varepsilon \to \varepsilon$-approximation)?

Lower semicomputable random reals

- $\sum a_i$ computable converging series with rational terms
- is $\alpha = \sum a_i$ computable ($\varepsilon \to \varepsilon$-approximation)?
- not necessarily (Specker example)

Lower semicomputable random reals

- $\sum a_i$ computable converging series with rational terms
- is $\alpha = \sum a_i$ computable ($\varepsilon \to \varepsilon$-approximation)?
- not necessarily (Specker example)
- lower semicomputable reals

Lower semicomputable random reals

- $\sum a_i$ computable converging series with rational terms
- is $\alpha = \sum a_i$ computable ($\varepsilon \to \varepsilon$-approximation)?
- not necessarily (Specker example)
- lower semicomputable reals
- Solovay classification: $\alpha \preceq \beta$ if ε-approximation to β can be effectively converted to $O(\varepsilon)$-approximation to α

Lower semicomputable random reals

- $\sum a_i$ computable converging series with rational terms
- is $\alpha = \sum a_i$ computable ($\varepsilon \to \varepsilon$-approximation)?
- not necessarily (Specker example)
- lower semicomputable reals
- Solovay classification: $\alpha \preceq \beta$ if ε-approximation to β can be effectively converted to $O(\varepsilon)$-approximation to α
- There are maximal elements

Lower semicomputable random reals

- $\sum a_i$ computable converging series with rational terms
- is $\alpha = \sum a_i$ computable ($\varepsilon \to \varepsilon$-approximation)?
- not necessarily (Specker example)
- lower semicomputable reals
- Solovay classification: $\alpha \preceq \beta$ if ε-approximation to β can be effectively converted to $O(\varepsilon)$-approximation to α
- There are maximal elements = random lower semicomputable reals

Lower semicomputable random reals

- $\sum a_i$ computable converging series with rational terms
- is $\alpha = \sum a_i$ computable ($\varepsilon \to \varepsilon$-approximation)?
- not necessarily (Specker example)
- lower semicomputable reals
- Solovay classification: $\alpha \preceq \beta$ if ε-approximation to β can be effectively converted to $O(\varepsilon)$-approximation to α
- There are maximal elements = random lower semicomputable reals
- = slowly converging series

Lower semicomputable random reals

- $\sum a_i$ computable converging series with rational terms
- is $\alpha = \sum a_i$ computable ($\varepsilon \to \varepsilon$-approximation)?
- not necessarily (Specker example)
- lower semicomputable reals
- Solovay classification: $\alpha \preceq \beta$ if ε-approximation to β can be effectively converted to $O(\varepsilon)$-approximation to α
- There are maximal elements = random lower semicomputable reals
- = slowly converging series
- modulus of convergence: $\varepsilon \mapsto N(\varepsilon)$ = how many terms are needed for ε-precision

Lower semicomputable random reals

- $\sum a_i$ computable converging series with rational terms
- is $\alpha = \sum a_i$ computable ($\varepsilon \to \varepsilon$-approximation)?
- not necessarily (Specker example)
- lower semicomputable reals
- Solovay classification: $\alpha \preceq \beta$ if ε-approximation to β can be effectively converted to $O(\varepsilon)$-approximation to α
- There are maximal elements = random lower semicomputable reals
- = slowly converging series
- modulus of convergence: $\varepsilon \mapsto N(\varepsilon)$ = how many terms are needed for ε-precision
- maximal elements: $N(2^{-n}) > BP(n - O(1))$, where $BP(k)$ is the maximal integer whose prefix complexity is k or less.

Lovasz local lemma: constructive proof

Lovasz local lemma: constructive proof

- CNF: $(a \wedge \neg b \wedge \ldots) \vee (\neg c \wedge e \wedge \ldots) \vee \ldots$

Lovasz local lemma: constructive proof

- CNF: $(a \wedge \neg b \wedge \ldots) \vee (\neg c \wedge e \wedge \ldots) \vee \ldots$
- neighbors: clauses having common variables

Lovasz local lemma: constructive proof

- CNF: $(a \wedge \neg b \wedge \ldots) \vee (\neg c \wedge e \wedge \ldots) \vee \ldots$
- neighbors: clauses having common variables
- several clauses with k literals in each

Lovasz local lemma: constructive proof

- CNF: $(a \wedge \neg b \wedge \ldots) \vee (\neg c \wedge e \wedge \ldots) \vee \ldots$
- neighbors: clauses having common variables
- several clauses with k literals in each
- each clause has $o(2^k)$ neighbors

Lovasz local lemma: constructive proof

- CNF: $(a \wedge \neg b \wedge \ldots) \vee (\neg c \wedge e \wedge \ldots) \vee \ldots$
- neighbors: clauses having common variables
- several clauses with k literals in each
- each clause has $o(2^k)$ neighbors
- $\Rightarrow$ CNF is satisfiable

Lovasz local lemma: constructive proof

- CNF: $(a \wedge \neg b \wedge \ldots) \vee (\neg c \wedge e \wedge \ldots) \vee \ldots$
- neighbors: clauses having common variables
- several clauses with k literals in each
- each clause has $o(2^k)$ neighbors
- $\Rightarrow$ CNF is satisfiable
- Non-constructive proof: lower bound for probability, Lovasz local lemma

Lovasz local lemma: constructive proof

- CNF: $(a \wedge \neg b \wedge \ldots) \vee (\neg c \wedge e \wedge \ldots) \vee \ldots$
- neighbors: clauses having common variables
- several clauses with k literals in each
- each clause has $o(2^k)$ neighbors
- $\Rightarrow$ CNF is satisfiable
- Non-constructive proof: lower bound for probability, Lovasz local lemma
- Naïve algorithm: just resample false clause (while they exist)

Lovasz local lemma: constructive proof

- CNF: $(a \wedge \neg b \wedge \ldots) \vee (\neg c \wedge e \wedge \ldots) \vee \ldots$
- neighbors: clauses having common variables
- several clauses with k literals in each
- each clause has $o(2^k)$ neighbors
- $\Rightarrow$ CNF is satisfiable
- Non-constructive proof: lower bound for probability, Lovasz local lemma
- Naïve algorithm: just resample false clause (while they exist)
- Recent breakthrough (Moser): this algorithm with high probability terminates quickly

Lovasz local lemma: constructive proof

- CNF: $(a \wedge \neg b \wedge \ldots) \vee (\neg c \wedge e \wedge \ldots) \vee \ldots$
- neighbors: clauses having common variables
- several clauses with k literals in each
- each clause has $o(2^k)$ neighbors
- $\Rightarrow$ CNF is satisfiable
- Non-constructive proof: lower bound for probability, Lovasz local lemma
- Naïve algorithm: just resample false clause (while they exist)
- Recent breakthrough (Moser): this algorithm with high probability terminates quickly
- Explanation: if not, the sequence of resampled clauses would encode the random bits used in resampling making them compressible

Berry, Gödel, Chaitin, Raz

Berry, Gödel, Chaitin, Raz

- There are only finitely many strings of complexity $< n$

Berry, Gödel, Chaitin, Raz

- There are only finitely many strings of complexity $< n$
- for all strings (except finitely many ones) x the statement $K(x) > n$ is true

Berry, Gödel, Chaitin, Raz

- There are only finitely many strings of complexity $< n$
- for all strings (except finitely many ones) x the statement $K(x) > n$ is true
- Can all true statements of this form be provable?

Berry, Gödel, Chaitin, Raz

- There are only finitely many strings of complexity $< n$
- for all strings (except finitely many ones) x the statement $K(x) > n$ is true
- Can all true statements of this form be provable?
- No, otherwise we could effectively generate string of complexity $> n$ by enumerating all proofs

Berry, Gödel, Chaitin, Raz

- There are only finitely many strings of complexity $< n$
- for all strings (except finitely many ones) x the statement $K(x) > n$ is true
- Can all true statements of this form be provable?
- No, otherwise we could effectively generate string of complexity $> n$ by enumerating all proofs
- and get Berry's paradox: the first provable statement $C(x) > n$ for given n gives some x of complexity $> n$ that can be described by $O(\log n)$ bits

Berry, Gödel, Chaitin, Raz

- There are only finitely many strings of complexity $< n$
- for all strings (except finitely many ones) x the statement $K(x) > n$ is true
- Can all true statements of this form be provable?
- No, otherwise we could effectively generate string of complexity $> n$ by enumerating all proofs
- and get Berry's paradox: the first provable statement $C(x) > n$ for given n gives some x of complexity $> n$ that can be described by $O(\log n)$ bits
- (Gödel theorem in Chaitin form): There are only finitely many n such that $C(x) > n$ is provable for some x.

Berry, Gödel, Chaitin, Raz

- There are only finitely many strings of complexity $< n$
- for all strings (except finitely many ones) x the statement $K(x) > n$ is true
- Can all true statements of this form be provable?
- No, otherwise we could effectively generate string of complexity $> n$ by enumerating all proofs
- and get Berry's paradox: the first provable statement $C(x) > n$ for given n gives some x of complexity $> n$ that can be described by $O(\log n)$ bits
- (Gödel theorem in Chaitin form): There are only finitely many n such that $C(x) > n$ is provable for some x. (Note that $\exists x\, C(x) > n$ is always provable!)

Berry, Gödel, Chaitin, Raz

- There are only finitely many strings of complexity $< n$
- for all strings (except finitely many ones) x the statement $K(x) > n$ is true
- Can all true statements of this form be provable?
- No, otherwise we could effectively generate string of complexity $> n$ by enumerating all proofs
- and get Berry's paradox: the first provable statement $C(x) > n$ for given n gives some x of complexity $> n$ that can be described by $O(\log n)$ bits
- (Gödel theorem in Chaitin form): There are only finitely many n such that $C(x) > n$ is provable for some x. (Note that $\exists x\, C(x) > n$ is always provable!)
- (Gödel second theorem, Kritchman–Raz proof): the "unexpected test paradox" (a test will be given next week but it won't be known before the day of the test)

13th Hilbert's problem

13th Hilbert's problem

- Function of ≥ 3 variables: e.g., solution of a polynomial of degree 7 as function of its coefficients

13th Hilbert's problem

- Function of ≥ 3 variables: e.g., solution of a polynomial of degree 7 as function of its coefficients
- Is it possible to represent this function as a composition of functions of at most 2 variables?

13th Hilbert's problem

- Function of ≥ 3 variables: e.g., solution of a polynomial of degree 7 as function of its coefficients
- Is it possible to represent this function as a composition of functions of at most 2 variables?
- Yes, with weird functions (Cantor)

13th Hilbert's problem

- Function of ≥ 3 variables: e.g., solution of a polynomial of degree 7 as function of its coefficients
- Is it possible to represent this function as a composition of functions of at most 2 variables?
- Yes, with weird functions (Cantor)
- Yes, even with continuous functions (Kolmogorov, Arnold)

13th Hilbert's problem

- Function of ≥ 3 variables: e.g., solution of a polynomial of degree 7 as function of its coefficients
- Is it possible to represent this function as a composition of functions of at most 2 variables?
- Yes, with weird functions (Cantor)
- Yes, even with continuous functions (Kolmogorov, Arnold)
- Circuit version: explicit function $\mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}^n$ (polynomial circuit size?) that cannot be represented as composition of $O(1)$ functions $\mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}^n$. Not known.

13th Hilbert's problem

- Function of ≥ 3 variables: e.g., solution of a polynomial of degree 7 as function of its coefficients
- Is it possible to represent this function as a composition of functions of at most 2 variables?
- Yes, with weird functions (Cantor)
- Yes, even with continuous functions (Kolmogorov, Arnold)
- Circuit version: explicit function $\mathbb{B}^n \times \mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}^n$ (polynomial circuit size?) that cannot be represented as composition of $O(1)$ functions $\mathbb{B}^n \times \mathbb{B}^n \to \mathbb{B}^n$. Not known.
- Kolmogorov complexity version: we have three strings a, b, c on a blackboard. It is allowed to write (add) a new string if it simple relative to two of the strings on the board. Which strings we can obtain in $O(1)$ steps? Only strings of small complexity relative to a, b, c, but not all of them (for random a, b, c)

Secret sharing

Secret sharing

- secret s and three people; any two are able to reconstruct the secret working together; but each one in isolation has no information about it

Secret sharing

- secret s and three people; any two are able to reconstruct the secret working together; but each one in isolation has no information about it
- assume $s \in \mathbb{F}$ (a field); take random a and tell the people a, $a + s$ and $a + 2s$ (assuming $2 \neq 0$ in $\mathbb{F}$)

Secret sharing

- secret s and three people; any two are able to reconstruct the secret working together; but each one in isolation has no information about it
- assume $s \in \mathbb{F}$ (a field); take random a and tell the people a, $a + s$ and $a + 2s$ (assuming $2 \neq 0$ in $\mathbb{F}$)
- other secret sharing schemes; how long should be secrets (not understood)

Secret sharing

- secret s and three people; any two are able to reconstruct the secret working together; but each one in isolation has no information about it
- assume $s \in \mathbb{F}$ (a field); take random a and tell the people a, $a + s$ and $a + 2s$ (assuming $2 \neq 0$ in $\mathbb{F}$)
- other secret sharing schemes; how long should be secrets (not understood)
- Kolmogorov complexity settings: for a given s find a, b, c such that

$$C(s|a), C(s|b), C(s|c) \approx C(s);\ C(s|a,b), C(s|a,c), C(s|b,c) \approx 0$$

Secret sharing

- secret s and three people; any two are able to reconstruct the secret working together; but each one in isolation has no information about it
- assume $s \in \mathbb{F}$ (a field); take random a and tell the people a, $a + s$ and $a + 2s$ (assuming $2 \neq 0$ in $\mathbb{F}$)
- other secret sharing schemes; how long should be secrets (not understood)
- Kolmogorov complexity settings: for a given s find a, b, c such that

$$C(s|a), C(s|b), C(s|c) \approx C(s);\ C(s|a,b), C(s|a,c), C(s|b,c) \approx 0$$

- Some relation between Kolmogorov and traditional settings (Romashchenko, Kaced)

Quasi-cryptography

Quasi-cryptography

- Alice has some information a

Quasi-cryptography

- Alice has some information a
- Bob wants to let her know some b

Quasi-cryptography

- Alice has some information a
- Bob wants to let her know some b
- by sending some message f

Quasi-cryptography

- Alice has some information a
- Bob wants to let her know some b
- by sending some message f
- in such a way that Eve gets minimal information about b

Quasi-cryptography

- Alice has some information a
- Bob wants to let her know some b
- by sending some message f
- in such a way that Eve gets minimal information about b
- Formally: for given a, b find f such that $C(b|a,f) \approx 0$ and $C(b|f) \to \max$.

Quasi-cryptography

- Alice has some information a
- Bob wants to let her know some b
- by sending some message f
- in such a way that Eve gets minimal information about b
- Formally: for given a, b find f such that $C(b|a,f) \approx 0$ and $C(b|f) \to \max$.
- Theorem (Muchnik): it is always possible to have $C(b|f) \approx \min(C(b), C(a))$

Quasi-cryptography

- Alice has some information a
- Bob wants to let her know some b
- by sending some message f
- in such a way that Eve gets minimal information about b
- Formally: for given a, b find f such that $C(b|a,f) \approx 0$ and $C(b|f) \to \max$.
- Theorem (Muchnik): it is always possible to have $C(b|f) \approx \min(C(b), C(a))$
- Full version: Eve knows some c and we want to send message of a minimal possible length $C(b|a)$

Andrej Muchnik (1958-2007)