Improving the Space-Bounded Version of Muchnik's Conditional Complexity Theorem via "Naive" Derandomization

Daniil Musatov[1]

[1]Lomonosov Moscow State University,

St. Petersburg, June 14th, 2011

Some disclaimers

- This talk is in some sense a continuation of the previous one

Some disclaimers

- This talk is in some sense a continuation of the previous one
- A similar technique is used to obtain a different result

Some disclaimers

- This talk is in some sense a continuation of the previous one
- A similar technique is used to obtain a different result
- Why "naive"?

Some disclaimers

- This talk is in some sense a continuation of the previous one
- A similar technique is used to obtain a different result
- Why "naive"? Because we simply replace a random object by a pseudo-random one and it still does the job.

Kolmogorov complexity

- Kolmogorov complexity $C(a|b)$ of a string a conditional to b is the minimal length of a program p that produces a given b, i.e., $p(b) = a$

Kolmogorov complexity

- Kolmogorov complexity $C(a|b)$ of a string a conditional to b is the minimal length of a program p that produces a given b, i.e., $p(b) = a$
- Space-bounded version: $C^s(a|b)$ is the minimal length of a program p such that $p(b) = a$

Kolmogorov complexity

- Kolmogorov complexity $C(a|b)$ of a string a conditional to b is the minimal length of a program p that produces a given b, i.e., $p(b) = a$
- Space-bounded version: $C^s(a|b)$ is the minimal length of a program p such that $p(b) = a$ and the computation of $p(b)$ performs in space s.

Muchnik's theorem

- Muchnik's theorem (TCS'2002): For any a and b of length n there exists p of length $C(a|b) + O(\log n)$ such that $p(b) = a$

Muchnik's theorem

- Muchnik's theorem (TCS'2002): For any a and b of length n there exists p of length $C(a|b) + O(\log n)$ such that $p(b) = a$ and $C(p|a) = O(\log n)$.

Muchnik's theorem

- Muchnik's theorem (TCS'2002): For any a and b of length n there exists p of length $C(a|b) + O(\log n)$ such that $p(b) = a$ and $C(p|a) = O(\log n)$.
- Space-bounded version (M., Romashchenko, Shen, CSR'2009, ToCS'2011): For any a and b of length n and for any s there exists p of length $C^s(a|b) + O(\log^3 n)$ such that:

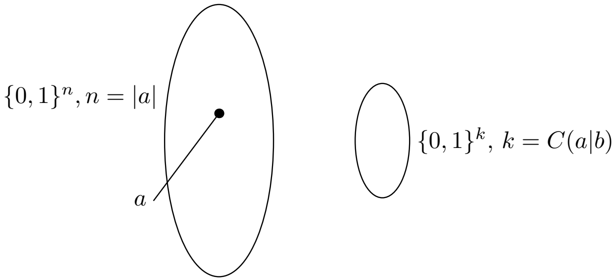
Muchnik's theorem

- Muchnik's theorem (TCS'2002): For any a and b of length n there exists p of length $C(a|b) + O(\log n)$ such that $p(b) = a$ and $C(p|a) = O(\log n)$.
- Space-bounded version (M., Romashchenko, Shen, CSR'2009, ToCS'2011): For any a and b of length n and for any s there exists p of length $C^s(a|b) + O(\log^3 n)$ such that:
  - $p(b) = a$;
  - the computation of $p(b)$ performs in space $O(s) + \mathrm{poly}(n)$
  - and $C^{\mathrm{poly}(n)}(p|a) = O(\log^3 n)$

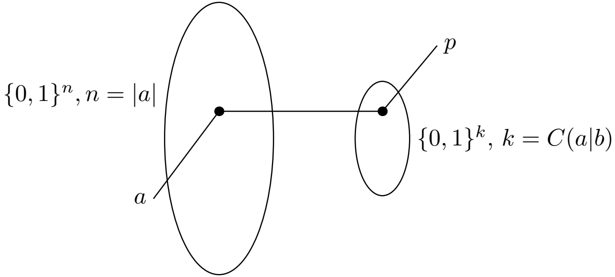
Muchnik's theorem

- Muchnik's theorem (TCS'2002): For any a and b of length n there exists p of length $C(a|b) + O(\log n)$ such that $p(b) = a$ and $C(p|a) = O(\log n)$.
- Space-bounded version (M., Romashchenko, Shen, CSR'2009, ToCS'2011): For any a and b of length n and for any s there exists p of length $C^s(a|b) + O(\log^3 n)$ such that:
  - $p(b) = a$;
  - the computation of $p(b)$ performs in space $O(s) + \mathrm{poly}(n)$
  - and $C^{\mathrm{poly}(n)}(p|a) = O(\log^3 n)$
- In current work we get rid of polylogarithmic terms and make them again logarithmic

Proof of Muchnik's theorem: fingerprints

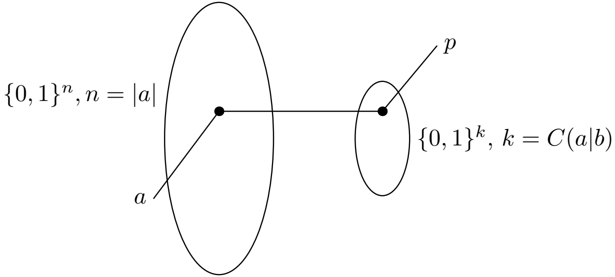


Proof of Muchnik's theorem: fingerprints



p is a *fingerprint* of a

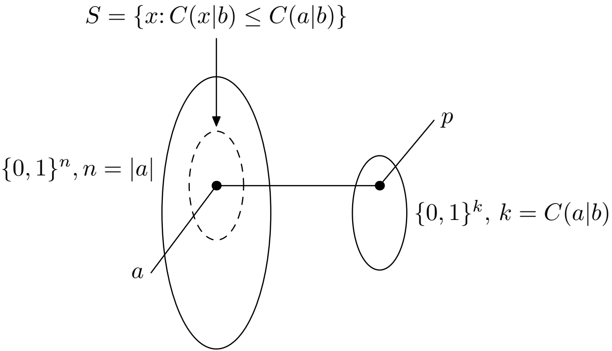
Proof of Muchnik's theorem: fingerprints



p is a *fingerprint* of *a*

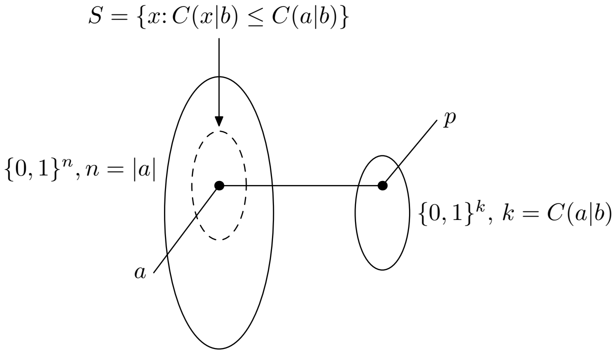
For $C(p|a)$ to be small, there should be few fingerprints for each a

Proof of Muchnik's theorem: fingerprints



p is a *fingerprint* of a
For $C(p|a)$ to be small, there should be few fingerprints for each a
For $C(a|p,b)$ to be small,

Proof of Muchnik's theorem: fingerprints



p is a *fingerprint* of a

For $C(p|a)$ to be small, there should be few fingerprints for each a

For $C(a|p, b)$ to be small, there should be few preimages of p in S

How to find fingerprints

- There is an underlying bipartite graph $G = (L, R, E)$ with $|L| = 2^n$ and $|R| = 2^k$

How to find fingerprints

- There is an underlying bipartite graph $G = (L, R, E)$ with $|L| = 2^n$ and $|R| = 2^k$
- Each left-part vertex has constant degree D

How to find fingerprints

- There is an underlying bipartite graph $G = (L, R, E)$ with $|L| = 2^n$ and $|R| = 2^k$
- Each left-part vertex has constant degree D
- The fingerprint p is chosen among neighbors of a

How to find fingerprints

- There is an underlying bipartite graph $G = (L, R, E)$ with $|L| = 2^n$ and $|R| = 2^k$
- Each left-part vertex has constant degree D
- The fingerprint p is chosen among neighbors of a
  - This guarantees that $C(p|a, G) = O(\log D)$

How to find fingerprints

- There is an underlying bipartite graph $G = (L, R, E)$ with $|L| = 2^n$ and $|R| = 2^k$
- Each left-part vertex has constant degree D
- The fingerprint p is chosen among neighbors of a
  - This guarantees that $C(p|a, G) = O(\log D)$
- The statement that $C(a|p, b, G)$ is small for some p follows from some enumerable graph property (see later)

How to find fingerprints

- There is an underlying bipartite graph $G = (L, R, E)$ with $|L| = 2^n$ and $|R| = 2^k$
- Each left-part vertex has constant degree D
- The fingerprint p is chosen among neighbors of a
  - This guarantees that $C(p|a, G) = O(\log D)$
- The statement that $C(a|p, b, G)$ is small for some p follows from some enumerable graph property (see later)
- It is proven by the probabilistic method that this property is non-empty

How to find fingerprints

- There is an underlying bipartite graph $G = (L, R, E)$ with $|L| = 2^n$ and $|R| = 2^k$
- Each left-part vertex has constant degree D
- The fingerprint p is chosen among neighbors of a
  - This guarantees that $C(p|a, G) = O(\log D)$
- The statement that $C(a|p, b, G)$ is small for some p follows from some enumerable graph property (see later)
- It is proven by the probabilistic method that this property is non-empty
- Hence, the first graph in the enumeration has small complexity

How to find fingerprints

- There is an underlying bipartite graph $G = (L, R, E)$ with $|L| = 2^n$ and $|R| = 2^k$
- Each left-part vertex has constant degree D
- The fingerprint p is chosen among neighbors of a
  - This guarantees that $C(p|a, G) = O(\log D)$
- The statement that $C(a|p, b, G)$ is small for some p follows from some enumerable graph property (see later)
- It is proven by the probabilistic method that this property is non-empty
- Hence, the first graph in the enumeration has small complexity
- Hence, $C(p|a)$ and $C(a|p, b)$ are also small

What graph properties do we need?

Some graph properties leading to Muchnik's theorem:

- (Muchnik) Expander-like property

What graph properties do we need?

Some graph properties leading to Muchnik's theorem:

- (Muchnik) Expander-like property
- (MRS) Possibility of online matching

What graph properties do we need?

Some graph properties leading to Muchnik's theorem:

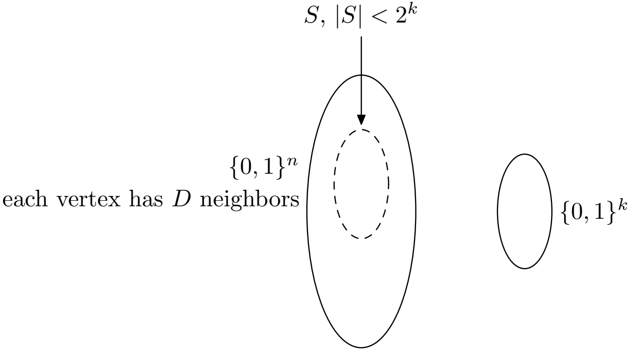
- (Muchnik) Expander-like property
- (MRS) Possibility of online matching
- (MRS) Extractor

What graph properties do we need?

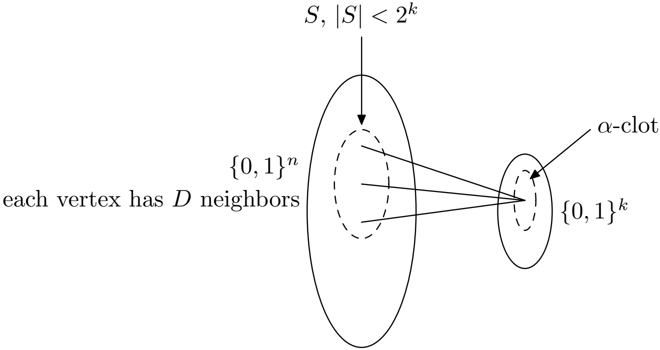
Some graph properties leading to Muchnik's theorem:

- (Muchnik) Expander-like property
- (MRS) Possibility of online matching
- (MRS) Extractor
- (This paper) "Low-congesting"

The essential property: low-congesting graph

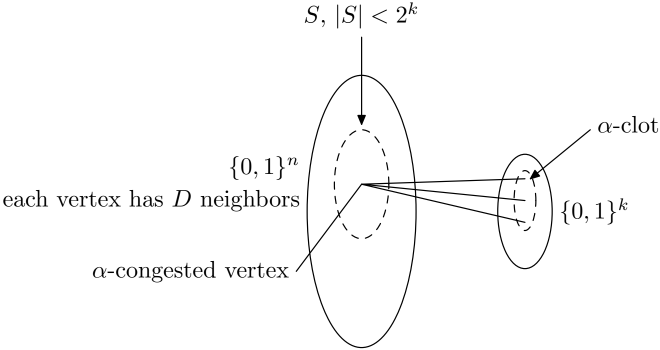


The essential property: low-congesting graph



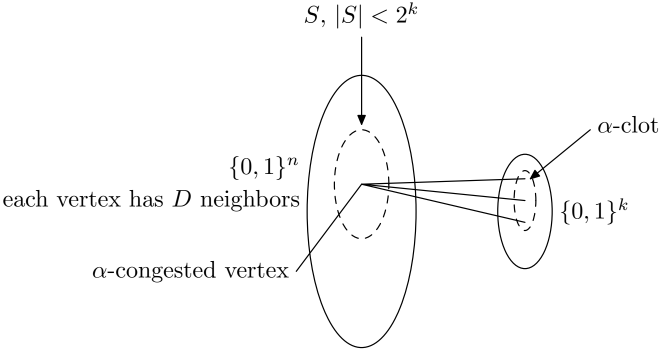
- α-clot for S is the set of all vertices having more than αD neighbors in S.

The essential property: low-congesting graph



- α-clot for S is the set of all vertices having more than αD neighbors in S.
- A vertex is α-congested if all its neighbors lie in α-clot.

The essential property: low-congesting graph



- α-clot for S is the set of all vertices having more than αD neighbors in S.
- A vertex is α-congested if all its neighbors lie in α-clot.
- The set S is $(\alpha,\, \beta)$-low-congested if it contains less than βK α-congested vertices.

The essential property: low-congesting graph

- α-clot for S is the set of all vertices having more than αD neighbors in S.
- A vertex is α-congested if all its neighbors lie in α-clot.
- The set S is (α, β)-low-congested if it contains less than βK α-congested vertices.
- We call a set *relevant* if it has the form $\{x | C^s(x|b) < k\}$.
- We call a graph (α, β)-low-congesting if all relevant sets are (α, β)-low-congested.

How to get a low-congesting graph

- A random graph with certain parameters is an extractor with positive probability.

How to get a low-congesting graph

- A random graph with certain parameters is an extractor with positive probability.
- **Lemma.** (Buhrman, Fortnow, Laplante '2002) In an (k, ϵ)-extractor graph any S is (2, 2ϵ)-low-congested.
- Hence, in an extractor any relevant set is low-congested and the graph itself is low-congesting.

How to get a low-congesting graph

- A random graph with certain parameters is an extractor with positive probability.
- **Lemma.** (Buhrman, Fortnow, Laplante '2002) In an (k, ϵ)-extractor graph any S is (2, 2ϵ)-low-congested.
- Hence, in an extractor any relevant set is low-congested and the graph itself is low-congesting.
- But even the description of the graph is exponential in size, so we cannot find it in polynomial space.

How to get a low-congesting graph

- A random graph with certain parameters is an extractor with positive probability.
- **Lemma.** (Buhrman, Fortnow, Laplante '2002) In an (k, ϵ)-extractor graph any S is $(2, 2\epsilon)$-low-congested.
- Hence, in an extractor any relevant set is low-congested and the graph itself is low-congesting.
- But even the description of the graph is exponential in size, so we cannot find it in polynomial space.
- **Central idea:** replace a random graph by a pseudorandom one
- To make this idea work, we need:

How to get a low-congesting graph

- A random graph with certain parameters is an extractor with positive probability.
- **Lemma.** (Buhrman, Fortnow, Laplante '2002) In an (k, ϵ)-extractor graph any S is (2, 2ϵ)-low-congested.
- Hence, in an extractor any relevant set is low-congested and the graph itself is low-congesting.
- But even the description of the graph is exponential in size, so we cannot find it in polynomial space.
- **Central idea:** replace a random graph by a pseudorandom one
- To make this idea work, we need:
    - to prove that a pseudorandom graph is low-congesting with positive probability

How to get a low-congesting graph

- A random graph with certain parameters is an extractor with positive probability.
- **Lemma.** (Buhrman, Fortnow, Laplante '2002) In an (k, ϵ)-extractor graph any S is (2, 2ϵ)-low-congested.
- Hence, in an extractor any relevant set is low-congested and the graph itself is low-congesting.
- But even the description of the graph is exponential in size, so we cannot find it in polynomial space.
- **Central idea:** replace a random graph by a pseudorandom one
- To make this idea work, we need:
  - to prove that a pseudorandom graph is low-congesting with positive probability
  - to show that the seed this graph is generated from may be found in polynomial space

Why a pseudorandom graph fits

- We use the Nisan pseudorandom generator with polynomial seed length and exponential output.

Why a pseudorandom graph fits

- We use the Nisan pseudorandom generator with polynomial seed length and exponential output.
- It is well-known that this generator fools any circuit from AC^0

Why a pseudorandom graph fits

- We use the Nisan pseudorandom generator with polynomial seed length and exponential output.
- It is well-known that this generator fools any circuit from AC^0
  - The size of a fooled circuit may be exponential since the size of output is exponential.

Why a pseudorandom graph fits

- We use the Nisan pseudorandom generator with polynomial seed length and exponential output.
- It is well-known that this generator fools any circuit from AC^0
  - The size of a fooled circuit may be exponential since the size of output is exponential.
- We cannot check the low-congesting property literally, but using circuits for approximate counting we build a circuit C such that:

Why a pseudorandom graph fits

- We use the Nisan pseudorandom generator with polynomial seed length and exponential output.
- It is well-known that this generator fools any circuit from AC^0
  - The size of a fooled circuit may be exponential since the size of output is exponential.
- We cannot check the low-congesting property literally, but using circuits for approximate counting we build a circuit C such that:
  - If G is $(2, 2\epsilon)$-low-congesting then $C(G) = 1$;

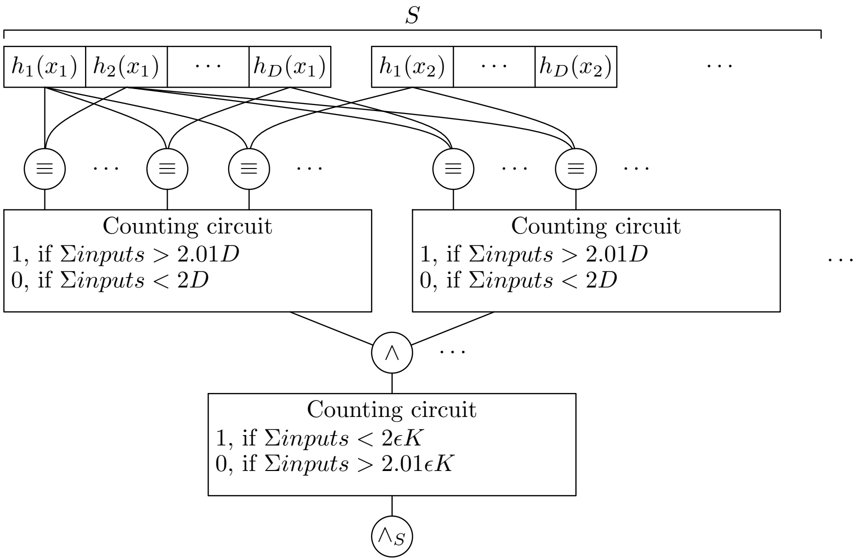
Why a pseudorandom graph fits

- We use the Nisan pseudorandom generator with polynomial seed length and exponential output.
- It is well-known that this generator fools any circuit from AC^0
  - The size of a fooled circuit may be exponential since the size of output is exponential.
- We cannot check the low-congesting property literally, but using circuits for approximate counting we build a circuit C such that:
  - If G is $(2, 2\epsilon)$-low-congesting then $C(G) = 1$;
  - If $C(G) = 1$ then G is $(2.01, 2.01\epsilon)$-low-congesting.

Why a pseudorandom graph fits

- We use the Nisan pseudorandom generator with polynomial seed length and exponential output.
- It is well-known that this generator fools any circuit from AC^0
  - The size of a fooled circuit may be exponential since the size of output is exponential.
- We cannot check the low-congesting property literally, but using circuits for approximate counting we build a circuit C such that:
  - If G is $(2, 2\epsilon)$-low-congesting then $C(G) = 1$;
  - If $C(G) = 1$ then G is $(2.01, 2.01\epsilon)$-low-congesting.
- This circuit accepts a random graph with sufficient probability, hence it does the same with a pseudorandom one.

The circuit


S
$h_1(x_1)$
$h_2(x_1)$
$\cdots$
$h_D(x_1)$
$h_1(x_2)$
$\cdots$
$h_D(x_2)$
$\cdots$
$\equiv$
Counting circuit
1, if $\Sigma inputs > 2.01D$
0, if $\Sigma inputs < 2D$
Counting circuit
1, if $\Sigma inputs > 2.01D$
0, if $\Sigma inputs < 2D$
$\wedge$
Counting circuit
1, if $\Sigma inputs < 2\epsilon K$
0, if $\Sigma inputs > 2.01\epsilon K$
$\wedge_S$

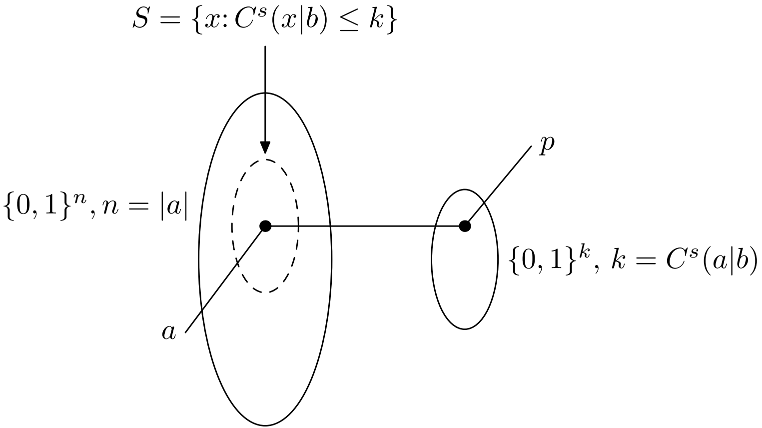
How to get Muchnik's theorem

- Firstly, we search for a good seed for the generator and fix it. This search needs only polynomial space.

How to get Muchnik's theorem

- Firstly, we search for a good seed for the generator and fix it. This search needs only polynomial space.
- If a is not congested in $\{x|C^s(x|b) < k\}$ then it has a neighbor p outside the clot.
- This p satisfies all requirements.

How to get Muchnik's theorem



How to get Muchnik's theorem

- Firstly, we search for a good seed for the generator and fix it. This search needs only polynomial space.
- If a is not congested in $\{x | C^s(x|b) < k\}$ then it has a neighbor p outside the clot.
- This p satisfies all requirements.
- If a *is* congested then we repeat the whole construction replacing the relevant sets by the sets of congested vertices in relevant sets.

How to get Muchnik's theorem

- Firstly, we search for a good seed for the generator and fix it. This search needs only polynomial space.
- If a is not congested in $\{x | C^s(x|b) < k\}$ then it has a neighbor p outside the clot.
- This p satisfies all requirements.
- If a *is* congested then we repeat the whole construction replacing the relevant sets by the sets of congested vertices in relevant sets.
- There may be several iterations but since the upper bound on the size of a relevant set decreases exponentially there is at most linear number of steps, hence all polynomial bounds remain.

The final formulation

For any a and b of length n and for any s there exists p of length $C^s(a|b) + O(\log\log s + \log n)$ such that:

- $p(b) = a$;
- the computation of $p(b)$ performs in space $O(s) + \text{poly}(n)$
- and $C^{O(s)+\text{poly}(n)}(p|a) = O(\log\log s + \log n)$

Summary of the technology

- Take some theorem about Kolmogorov complexity relying on the existence of some combinatorial object

Summary of the technology

- Take some theorem about Kolmogorov complexity relying on the existence of some combinatorial object
- Build a constant-depth circuit recognizing this object

Summary of the technology

- Take some theorem about Kolmogorov complexity relying on the existence of some combinatorial object
- Build a constant-depth circuit recognizing this object
- Replace a random object by a pseudorandom one

Summary of the technology

- Take some theorem about Kolmogorov complexity relying on the existence of some combinatorial object
- Build a constant-depth circuit recognizing this object
- Replace a random object by a pseudorandom one
- Obtain a space-bounded version of the theorem

Summary of the technology

- Take some theorem about Kolmogorov complexity relying on the existence of some combinatorial object
- Build a constant-depth circuit recognizing this object
- Replace a random object by a pseudorandom one
- Obtain a space-bounded version of the theorem
- ???????
- PROFIT

Thank you!

mailto:musatych@gmail.com
http://musatych.livejournal.com