

# Randomness as Circuit Complexity (and the Connection to Pseudorandomness)

Eric Allender\*

May 12, 2010

## Abstract

This material was generated for the book **Randomness through Computation**, edited by Hector Zenil. The format of the book calls for various contributors to give responses to five questions.

### Why were you initially drawn to the study of computation and randomness?

I started out my undergraduate studies majoring in theater. However, I also knew that I enjoyed math and computers, thanks to a terrific high school math teacher (Ed Rolenc) who installed a computer in one of the classrooms in my high school in Mount Pleasant, Iowa in the 1970's. Thus, when I was an undergrad at the University of Iowa and I decided to pick a second major that might make me more employable in case theater didn't work out, I picked Computer Science. My initial computing courses didn't really inspire me very much, and I continued to think of computing as a "reserve" career, until two things happened at more-or-less the same time: (1) I worked in summer theaters for a couple of summers, and I noticed that there were *incredibly talented* people who were working at the same undistinguished summer theaters where I was working, leading me to wonder how much impact I was likely to have in the field of theater. (2) I took my first courses in theoretical computer science (with Ted Baker and Don Alton), which opened my eyes to some of the fascinating open questions in the field. They gave me some encouragement to consider grad school, and thus (after taking a year off to "see the world" by working as a bellhop in Germany) I entered the doctoral program at Georgia Tech.

At that time, in the early-to-mid 1980's, Kolmogorov complexity was finding application in complexity theory. (I'm referring to the material that's now covered in the chapter called "The Incompressibility Method" in the standard text by Li and Vitányi [LV08].) Also, there were influential papers by Mike Sipser [Sip83] and Juris Hartmanis [Har83] that discussed resource-bounded Kolmogorov complexity. Also, the theory of pseudorandom generators was just getting underway, with the work of Yao [Yao82] and of Blum and Micali [BM84]. Thus Kolmogorov complexity and randomness were very much in the air.

Please recall that, at the time, most people believed that BPP and RP were likely to contain problems that could *not* be solved in *deterministic* polynomial time. (BPP is the class of problems that can be solved efficiently if we have access to randomness, and RP is the natural way to define a probabilistic subclass of NP.) I was struck by the observation that there was some tension between the popular conjectures relating deterministic, probabilistic, and nondeterministic computation. For example, if nondeterministic exponential time has no efficient deterministic simulation, it means

---

\*Department of Computer Science, Rutgers University. Supported in part by NSF Grants DMS-0652582, CCF-0830133, and CCF-0832787.

that there are sets in P that (for many input lengths) contain no strings of low resource-bounded Kolmogorov complexity. However, if efficient pseudorandom generators exist, then every set in P *must* contain strings of “low” resource-bounded Kolmogorov complexity, *if it* contains many strings of a given length. However, if one quantifies “low complexity” as meaning “complexity  $O(\log n)$ ,” then it would follow that  $RP=P$  (and now we know that this same hypothesis implies  $BPP=P$ ). So just what is going on, when one considers the resource-bounded Kolmogorov complexity of sets in P? Can an efficient computation say “yes” to a large number of strings, without accepting strings with low resource-bounded Kolmogorov complexity? My first STOC paper grew out of precisely these considerations [All89].

### What have we learned?

At a basic level, the most important lesson this field gives us is conceptual. Until the mathematical framework that we now call Kolmogorov Complexity was established, there was no meaningful way to talk about a given object being “random”. Now there is. And it is crucial that the key ingredient is the notion of *computability*. Until we grappled with computation, we could not understand randomness.

Starting from that flash of insight and a few simple definitions, the field has grown remarkably. You are asking me “What have we learned?” My initial reaction is to simply point you to a standard textbook such as the one by Li and Vitányi [LV08] or the soon-to-be-released volume by Downey and Hirschfeldt [DH10]. But instead, my answer will concentrate on some material that is not emphasized in these volumes.

For me, one of the most surprising and simple insights that has come out of the study of randomness recently, is the fact that there is a very natural and close connection between Kolmogorov complexity and circuit complexity.

At first blush, these topics seem to have nothing to do with each other, for several reasons:

1. *Kolmogorov complexity measures the complexity of strings, while circuit complexity measures the complexity of functions.* This, of course, is no significant difference at all. Any string  $x$  of length  $m$  can be viewed as the initial part of the truth table of a function  $f_x$  having size  $2^n < 2m$ . Thus we can define  $\text{Size}(x)$  to be the size of the smallest circuit computing  $f_x$ . Viewed in this way, Kolmogorov complexity and circuit complexity each measure the complexity of strings.

2. *Kolmogorov complexity is not computable, in stark contrast to circuit complexity.* This, on the other hand, would seem to present an insurmountable barrier to making any meaningful connection between Kolmogorov complexity and circuit complexity. The way around this barrier is to employ one of the oldest tricks in the toolkit of complexity and computability: oracles.

The function  $\text{Size}(x)$  gives the size of the smallest circuit computing  $f_x$ , where the circuit is made up of the usual circuit components: AND and OR gates. But there is a natural and well-studied notion of providing oracle access to a circuit. An oracle gate for a set  $A$  has some number of wires (say,  $m$ ) as input, and produces as output the answer to the question “*Is  $z$  in  $A$ ?*” where  $z$  is the  $m$ -bit string that is fed into the input wires of the gate. We define  $\text{Size}^A(x)$  to be the size of the smallest circuit having oracle gates for  $A$  that computes  $f_x$ .

It’s not too hard to show that both  $K(x)$  and  $C(x)$  (the usual Kolmogorov complexity measures) are polynomially-related to  $\text{Size}^H(x)$ , where  $H$  is the halting problem (or any other set that is complete for the computably-enumerable sets under polynomial-time reductions) [ABK<sup>+</sup>06b].

Similarly,  $\text{Kt}(x)$  is polynomially-related to  $\text{Size}^A(x)$ , where  $A$  is any set complete for exponential-time. (Here,  $\text{Kt}$  is a time-bounded notion of Kolmogorov complexity defined by Leonid Levin [Lev84].) In a paper that I wrote with Harry Buhrman, Michal Koucký, Dieter van Melkebeek, and Detlef Ronneburger [ABK<sup>+</sup>06b], we defined another notion of time-bounded Kolmogorov complexity in the style of Levin’s definition, that is polynomially-related to  $\text{Size}(x)$  (with no oracles). Thus we can see that the measures that are central to the studies of complexity and to the study

of randomness are, in fact, reflections of each other.

For me, the most exciting thing about this connection is that it enabled amazing techniques from the field of derandomization to be applied to questions about Kolmogorov complexity. Derandomization, and the study of pseudorandom generators, has been an incredibly active and productive field in the last few decades. Advances in that field have completely changed the way that many of us think about probabilistic computation.

As I've already mentioned, back when I was in grad school, most people in the field used to think that BPP and RP were probably strictly larger than P. Now, the situation is completely reversed. What happened to turn things around? Two things. There had previously been some interesting results, showing that the existence of hard-on-average one-way functions would imply somewhat fast deterministic simulation of probabilistic algorithms. But then Nisan and Wigderson came up with a new class of pseudorandom generators, based on the seemingly weaker assumption that there is a problem in exponential time that is hard on average in the sense of circuit complexity [NW94]. The real tide change came in 1997, when Impagliazzo and Wigderson weakened the assumption even further, by showing that  $BPP = P$  if there is any problem computable in exponential time that requires circuits of exponential size [IW97]. A number of additional important papers on derandomization followed soon thereafter.

But all of this exciting work showing how to eliminate the need for random bits in simulating BPP seemed divorced from the study of randomness in the sense of Kolmogorov complexity. By making use of the connection between Kolmogorov complexity and circuit complexity, we were able to bridge this divide, and present some interesting reductions. For instance, consider the set of random strings (using K or C complexity). Of course, the set of random strings is undecidable, and is Turing-equivalent to the Halting problem – but it is far from obvious how to make use of this set using an *efficient* reduction. We were able to show that PSPACE is poly-time Turing reducible to the set of random strings [ABK<sup>+</sup>06b], and NEXP is in NP relative to this set [ABK06a]. Even the Halting problem itself is efficiently reducible to the set of random strings – if one allows reductions computable by poly-size *circuits*, instead of poly-time machines [ABK<sup>+</sup>06b].

Similar techniques show that the set of random strings in the sense of Levin's Kt complexity is complete for EXP (under reductions computed by poly-size circuits, as well as under NP-Turing reductions).

### **What don't we know (yet)?**

Continuing in the same thread from the previous question, here are some annoying open questions:

- *Is the set of Kt-random strings in P?* Of course, the answer has to be “no”, or else EXP has poly-size circuits and the polynomial hierarchy collapses. But I know of no reason why it should be hard to *prove* unconditionally that this set is not in P.
- *Is EXP poly-time reducible to the set of random strings?* I think that it's reasonably likely that there is some complexity class larger than PSPACE that is poly-time reducible to the set of random strings, but it is far from clear to me that *every* decidable set should be poly-time reducible to the set of random strings. Right now, we cannot even rule out that the Halting problem is poly-time reducible to the set of random strings. The fact that there *is* such a reduction computable by poly-size *circuits* indicates some of the subtleties involved.

Perhaps the most glaring hole in our edifice of knowledge relating to randomness, is the fact that we have no real proof that randomness even *exists* in our universe. There is no proof that, say, quantum mechanical phenomena can be exploited in order to provide a satisfactory source of randomness at the macro level, and certainly there is no really satisfactory source of randomness

currently available for industrial applications that would dearly love to have a perfect source of randomness. For instance, in order for public-key cryptography to work at all, it is essential that secret keys (such as pairs of large primes for the RSA cryptosystem) be selected more-or-less uniformly and independently from a large space. If there is only a small set of likely keys, then the system is vulnerable to cryptanalysis. Thus there is a lot of money riding on the hope that keys are being generated at random – but it is not clear that there will *ever* be, in principle, a way to *prove* that a process is truly random. Even worse, it is not clear that one can ever fully disprove the theory of Laplacian Determinism – which means that we’ll never really know if randomness exists at all in our universe. So we’ll just have to keep muddling along, with regard to that question, relying on “faith” that enough randomness is available when it is needed.

Even here, the theory of derandomization has provided a number of useful tools. Trevisan [Tre01] showed that the Impagliazzo-Wigderson generator can be used as a “randomness extractor” that can take input from a “bad” source of randomness and produce a small list of samples (on a somewhat smaller space) that approximates what one would obtain if one had access to the uniform distribution on this space.

### **What are the most important open problems in the field?**

We can’t understand randomness without grappling with computation – and we can’t really claim to understand computation until the P vs NP question is settled. Thus I’ll start with the completely non-controversial choice of naming the P vs NP question as the most important open problem in the field.

And yet, for practical matters – as well as for the questions that drive the study of randomness – the P vs NP question is probably not as relevant as the question of the *circuit complexity* of various problems. To see what I mean by this, consider the problem faced by people who recommend what key size to use for various cryptosystems. Even if someone were able to prove  $P \neq NP$ , and were furthermore able to prove a superpolynomial run-time lower bound for the problem the cryptosystem is based on, it would still be *impossible* to choose a secure key size, if no circuit size lower bound were in hand. For instance, consider any problem that is complete for EXP; we know that every program solving such a problem must have a huge run-time for *large* inputs, but there is *absolutely* no way to say how large the inputs have to be, before the run-time must be large. There’s no guarantee that there can’t be a C++ program that can run on your laptop and solve the problem for inputs of 10,000 bits in a few seconds – precisely because we don’t know how to prove that EXP requires large circuits. In contrast, building on circuit lower bounds, Stockmeyer and Meyer showed that there are some natural and interesting problems that can’t be solved for inputs of size 400 by any circuit that will fit in the galaxy [SM02].

So *circuit size* is even more important to understand than program run-time. For the study of randomness, I’d have to say that one of the most important open questions is the question of whether there is any problem computable in time  $2^n$  (e.g., SAT) that requires circuits of size  $2^{\epsilon n}$  for some  $\epsilon > 0$ . This turns out to be *equivalent* to the question of whether pseudorandom generators exist (with certain parameters); this is surveyed nicely by Fortnow [For01].

### **What are the prospects for progress?**

I try to be optimistic. I wrote a survey recently [All08], outlining some of the approaches that have been proposed, to try to overcome the barriers that seem to block progress toward proving circuit lower bounds (and let me repeat that I think that circuit lower bounds are really the most important goal to strive for, if we want to understand randomness). In the survey, I tried to make the case that, although there is certainly a great deal of pessimism about the prospects for quick resolution of any of these problems (such as the P vs. NP problem), there is nonetheless some reason for hope.

More generally, the historical record shows that our understanding of this topic has evolved

significantly with each passing decade. Viewed in this light, I'm not only optimistic about the prospects for progress – I think that progress is all but inevitable.

## References

- [ABK06a] E. Allender, H. Buhrman, and M. Koucký. What can be efficiently reduced to the  $K$ -random strings? *Annals of Pure and Applied Logic*, 138:2–19, 2006.
- [ABK<sup>+</sup>06b] E. Allender, H. Buhrman, M. Koucký, D. van Melkebeek, and D. Ronneburger. Power from random strings. *SIAM Journal on Computing*, 35:1467–1493, 2006.
- [All89] E. Allender. Some consequences of the existence of pseudorandom generators. *Journal of Computer and System Sciences*, 39:101–124, 1989.
- [All08] E. Allender. Cracks in the defenses: Scouting out approaches on circuit lower bounds. In *Computer Science – Theory and Applications (CSR 2008)*, volume 5010 of *Lecture Notes in Computer Science*, pages 3–10, 2008.
- [BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM Journal on Computing*, 13:850–864, 1984.
- [DH10] R. Downey and D. Hirschfeldt. Algorithmic randomness and complexity. To be published., 2010.
- [For01] L. Fortnow. Comparing notions of full derandomization. In *IEEE Conference on Computational Complexity '01*, pages 28–34, 2001.
- [Har83] J. Hartmanis. Generalized Kolmogorov complexity and the structure of feasible computations (preliminary report). In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 439–445, 1983.
- [IW97] R. Impagliazzo and A. Wigderson.  $P = BPP$  if  $E$  requires exponential circuits: Derandomizing the XOR lemma. In *ACM Symposium on Theory of Computing (STOC) '97*, pages 220–229, 1997.
- [Lev84] L. Levin. Randomness conservation inequalities; information and independence in mathematical theories. *Information and Control*, 61:15–37, 1984.
- [LV08] M. Li and P. Vitányi. *Introduction to Kolmogorov Complexity and its Applications*. Springer, third edition, 2008.
- [NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *Journal of Computer and System Sciences*, 49:149–167, 1994.
- [Sip83] M. Sipser. A complexity theoretic approach to randomness. In *ACM Symposium on Theory of Computing (STOC)*, pages 330–335, 1983.
- [SM02] Larry J. Stockmeyer and Albert R. Meyer. Cosmological lower bound on the circuit complexity of a small problem in logic. *J. ACM*, 49(6):753–784, 2002.
- [Tre01] L. Trevisan. Construction of extractors using pseudo-random generators. *Journal of the ACM*, 48(4):860–879, 2001.

- [Yao82] A. Yao. Theory and application of trapdoor functions. In *IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.