

# This Week's Finds in Mathematical Physics (Week 226)

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math.research/2006-02/msg00046.html>

---

- *From:* [baez@xx](mailto:baez@xx) (John Baez)
  - *Date:* Fri, 10 Feb 2006 20:08:12 +0000 (UTC)
- 

Also available at <http://math.ucr.edu/home/baez/week226.html>

February 23, 2003

This Week's Finds in Mathematical Physics (Week 226)

John Baez

This month I'm hanging out at CIRM, the "Centre International de Recontres Mathematiques" near Marseille. It's like a little hotel with a classroom, library and computers, set at the edge of a forest that borders the region of limestone hills and cliffs called the Calanques on the coast of the Mediterranean. All they do here is hold conferences: guests come in, listen to math talks, and eat nice French meals prepared by the overworked staff.

This February they're having a conference on logic and computation, with a strong slant towards the use of diagrams:

1) Geometry of Computation 2006 (Geocal06), <http://iml.univ-mrs.fr/geocal06/>

The first week they had lots of talks on "higher-dimensional rewriting", where you use diagrams to draw ways of rewriting ways of rewriting ways of... rewriting strings of symbols. This is inherently connected to  $n$ -categories, since  $n$ -categories show up whenever you consider "processes that take a process and turn it into another process" – and iterate this notion until your eyes start bugging out.

The first week was pretty cool, and I hope to talk about it someday. The second week I've been recovering from the first week. But today I'll start with some astronomy pictures and some gossip I heard about cryptography, random sequences, and logic.

Let's zoom in on NGC 1097. Here's an amateur astronomer's photograph of this galaxy. It's a barred spiral galaxy that's colliding with a smaller elliptical galaxy called NGC 1097A:

2) Daniel Verschatse, Antilhue – Chile, NGC 1097 in Fornax, [http://www.astrosurf.com/antilhue/ngc\\_1097\\_in\\_fornax.htm](http://www.astrosurf.com/antilhue/ngc_1097_in_fornax.htm)

NGC 1097 is called a "Seyfert galaxy" because its center emits lots of radio waves, but not enough to be considered a full-fledged "quasar". As you can see, the center also emits visible light:

3) Spiral galaxy NGC 1097, European Southern Observatory,  
<http://www.eso.org/outreach/press-rel/pr-2004/pr-28-04.html#phot-35d-04>

Recently the VLT has gotten a really close look at the center of NGC 1097. VLT stands for "Very Large Telescope". It's actually four 8-meter telescopes and three smaller auxiliary ones, all of which can function as a single unit, making it the biggest telescope in the world. It's run by Europeans as part of the "European Southern Observatory" – but it's based on a mountain called Cerro Paranal in the driest part of the Atacama Desert in northern Chile, which makes for wonderful viewing conditions. They had to carry the enormous mirrors across rugged desert roads to build this observatory:

4) Mirror Transport, European Southern Observatory,  
<http://www.eso.org/outreach/press-rel/pr-1997/phot-35g-97.jpg>

but the view of the night sky up there makes it all worthwhile:

5) The Southern sky Above Paranal, European Southern Observatory,  
<http://www.eso.org/outreach/press-rel/pr-2005/images/phot-40b-05-normal.jpg>

and now they have everything they need to take advantage of that location:

6) The VLT Array on Paranal Mountain, European Southern Observatory,  
<http://www.eso.org/outreach/press-rel/pr-2000/phot-14a-00-normal.jpg>

So, what did they see when they looked at NGC 1097?

In the middle there seems to be a black hole, emitting radiation as filaments of gas and dust spiral in. This has caused a ring of new stars to form, which are ionizing the hydrogen in their vicinity:

7) Feeding the Monster, European Southern Observatory  
<http://www.eso.org/outreach/press-rel/pr-2005/phot-33-05.html>

This ring is about 5,500 light-years across! That sounds big, but the galaxy is 45 million light-years away, so this is a stunningly detailed photo. So, we can examine in great detail the process by which black holes eat galaxies.

Anyway....

Talking to the logicians and computer scientists here, I'm hearing lots of gossip that I don't usually get. For example, I just learned that in 2004 there was a successful collision attack on MD5.

Huh? Well, it sounds very technical, but it boils down to this: it means somebody took a function  $f$  that is known not to be one-to-one, and found  $x$  and  $y$  such that  $f(x) = f(y)$ .

You wouldn't think this would make news! But such functions, called "cryptographic hash functions", are used throughout the computer security business. The idea is that you can take any file and apply the hash function to compute a string of, say, 128 bits. It's supposed to be hard to find two files that give the same bit string. This lets you use the bit string as a kind of summary or "digest" of the file. It's also supposed to be hard to guess the contents of a file from its digest. This lets you show someone the digest of a file without giving away any secrets.

MD5 is a popular hash function invented by Ron Rivest in 1991. People use it for checking the integrity of files: first you compute the digest of a file, and then, when you send the file to someone, you also send the digest. If they're worried that the file has been corrupted or tampered with, they compute its digest and compare it to what you sent them.

People also use MD5 and other hash functions for things like digital fingerprinting and copy protection. To illustrate this I'll give a silly example that's easy to understand.

Suppose Alice proves Goldbach's conjecture in February and wants to take her time writing a nice paper about it... but she wants to be able to show she was the first to solve this problem, in case anyone else solves it while she's writing her paper.

To do this, she types up a quick note explaining her solution, feeds this file into a cryptographic hash function, and posts the resulting 128-bit string to the newsgroup sci.math in an article entitled "I PROVED GOLDBACH'S CONJECTURE!" A dated copy appears on Google for everyone to see.

Now, if Bob solves the problem in July while Alice is still writing up her solution, Alice can reveal her note. If anyone doubts she wrote her note back in February, they can apply the cryptographic hash function and check that yes, the result matches the bit string she posted on Google!

For this to work, it had better be hard for a nasty version of Alice to take Bob's solution and cook up some note summarizing it whose hash function equals some bit string she already posted.

So, it's very good if the hash function is resistant to a "preimage attack". A preimage attack is where for a given  $x$  you have a trick for finding  $y$  such that  $f(y) = f(x)$ .

Nobody has carried out a successful preimage attack on MD5. But, people have carried out a successful "collision attack". This is

## This Week's Finds in Mathematical Physics (Week 226)

where you can cook up pairs  $x, y$  such that  $f(x) = f(y)$ . This isn't as useful, since you don't have control over \*either\*  $x$  or  $y$ . But, there do exist fiendish schemes for conning people using collision attacks:

8) Magnus Daum and Stefan Lucks, Attacking hash functions by poisoned messages: "The Story of Alice and Her Boss", <http://www.cits.rub.de/MD5Collisions/>

On this webpage you can see a letter of recommendation for Alice and a letter granting her a security clearance which both have the same MD5 digest! It also explains how Alice could use these to do evil deeds.

For more on cryptographic hash functions and their woes, try these:

9) Cryptographic hash function, Wikipedia, [http://en.wikipedia.org/wiki/Cryptographic\\_hash](http://en.wikipedia.org/wiki/Cryptographic_hash)

Steve Friedl, An illustrated guide to cryptographic hashes, <http://www.unixwiz.net/techtips/iguide-crypto-hashes.html>

Now, if you're a mathematician, the whole idea of a cryptographic hash function may seem counterintuitive. Just for a change of pace, take another important cryptographic hash function, called SHA-1. This is a function that takes any string of up to  $2^{64}$  bits and gives a digest that's 160 bits long. So, it's just some function

$f: S \rightarrow T$

from a set  $S$  of size  $2^{2^{64} + 1}$  to a set  $T$  of size  $2^{160}$ .

The first set is vastly bigger. So, the function  $f$  must be far from one-to-one! So why in the world is anyone surprised, much less dismayed, when people find a way to generate two elements in the first set that map to the same element in the second?

One reason is that while the first set is much bigger than the second, the second is mighty big too!

Suppose you're trying to do a preimage attack. Someone hands you an element of  $T$  and asks you to find an element of  $S$  that maps to it. The brute-force approach, where you keep choosing elements of  $S$ , applying the function, and seeing if you get the desired element, will on average take about  $2^{160}$  tries. That's infeasible.

Note that the huge size of  $S$  is irrelevant here; what matters most is the size of  $T$ .

Or, suppose you're trying a brute-force collision attack. You start looking through elements of  $S$ , trying to find two that map to the same thing. On average it will take  $2^{80}$  tries – the square root of the size of  $T$ . That's a lot less, but still infeasible.

## This Week's Finds in Mathematical Physics (Week 226)

(Why so much less? This is called the "birthday paradox": it's a lot easier to find two people at a big party who share the same birthday than to find someone with the same birthday as the host.)

Of course, a smarter approach is to use your knowledge about the function  $f$  to help you find pairs of elements that map to the same thing. This is what Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu actually did in February 2005. Based on trials with watered-down versions of SHA-1, they argued that they could do a collision attack that would take only  $2^{69}$  tries instead of the expected  $2^{80}$ .

They didn't actually carry out this attack – with the computer power they had, it would have taken them 5 million years. But their theoretical argument was already enough to make people nervous!

And indeed, in August 2005, an improved version of their strategy reduced the necessary number of tries to  $2^{63}$  – in other words, reducing the time to just 78 thousand years, after only a few months of work.

So, people are getting wary of SHA-1. This is serious, because it's a widely used government standard. You can see the algorithm for this function here:

10) SHA hash functions, Wikipedia,  
[http://en.wikipedia.org/wiki/SHA\\_hash\\_functions](http://en.wikipedia.org/wiki/SHA_hash_functions)

People still hope that good hash functions exist. They hope that if a function  $f$  is cleverly chosen,  $f(x)$  will depend in a "seemingly random" way on  $x$ , so that given  $f(x)$ , it's hard to compute some  $y$  with  $f(y) = f(x)$ . People call this a "one-way function".

Now we're finally getting to the really interesting math. It takes work to make the concept of "one-way function" precise, but it can be done. For example, Chapter 2 of this book starts out by defining "strong" and "weak" one-way functions:

11) Oded Goldreich, Foundations of Cryptography, Cambridge U. Press, Cambridge, 2004. Older edition available at  
<http://www.wisdom.weizmann.ac.il/~oded/frag.html>

Since you can look them up and they're a bit gnarly, I won't give these definitions here. But \*roughly\*, a "strong one-way function" is a function  $f$  that satisfies two conditions:

A) You can compute  $f$  in "polynomial time": in other words, there's an algorithm that computes it in a number of steps bounded by some polynomial in the length of the input.

B) Given some input  $x$ , any polynomial-time probabilistic algorithm has a very low chance of finding  $y$  with  $f(y) = f(x)$ . Here a

## This Week's Finds in Mathematical Physics (Week 226)

"probabilistic algorithm" is just an algorithm equipped with access to a random number generator.

Condition B is related to the idea of a "preimage attack". We allow the attacker to use a probabilistic algorithm because it seems this can help them, and a strong one-way function should be able to resist even really nasty attacks.

Unfortunately, nobody has proved that a one-way function exists!

In fact, the existence of a one-way function would imply that "P does not equal NP". But, proving or disproving this claim is one of the most profound unsolved math problems around. If you settle it, you'll get a million dollars from the Clay Mathematics Institute:

11) Clay Mathematics Institute, P vs NP problem, [http://www.claymath.org/millennium/P\\_vs\\_NP/](http://www.claymath.org/millennium/P_vs_NP/)

But if you prove that P *does* equal NP, you might make more money by breaking cryptographic hash codes and setting yourself up as the Napoleon of crime.

The existence of one-way-functions is also closely related to the existence of "pseudorandom" sequences. These are sequences of numbers that "seem" random, but can be computed using deterministic algorithms – so they're not *really* random.

To see the relation, recall that a good cryptographic hash code shouldn't let you guess a message from its digest. So, for example, if my message is the EMPTY STRING – no message at all! – its digest should be a pseudorandom sequence.

For example, if we apply SHA-1 to the empty string we get the following 160 bits, written as 40 hexadecimal digits:

```
SHA1("") = "da39a3ee5e6b4b0d3255bfef95601890afd80709"
```

Seems mighty random to me! But of course it comes out the same every time.

Indeed, if you've ever seen a "random number generator" while messing with computers, it probably was a pseudorandom number generator. There are a few people who generate random numbers from physical phenomena like atmospheric noise. In fact, there's a website called random.org where you can get such numbers for free:

12) Random.org: random integer generator, <http://random.org/>

There's also a website that offers *nonrandom* numbers:

13) NoEntropy.net: your online source for truly deterministic numbers, <http://www.noentropy.net/>

But joking aside, it's a really tantalizing and famous problem to figure out what "random" means, and what it means for something to "seem" random even if it's the result of a deterministic process. These are huge and wonderful philosophico–physico–mathematical questions with serious practical implications. Much ink has been spilt regarding them, and I don't have the energy to discuss them carefully, so I'll just say some stuff to pique your curiosity, and then give you some references.

We can define a "random sequence" to be one that no algorithm can guess with a success rate better than chance would dictate. By virtue of this definition, no algorithm can generate truly random sequences. It's easy to prove that most sequences are random – but it's also easy to prove that you can never exhibit any one \*particular\* sequence and prove it's random! Chaitin has given a marvelous definition of a particular random sequence of bits called Omega using the fact that no algorithm can decide which Turing machines halt... but this random sequence is uncomputable, so you can't really "exhibit" it:

14) Gregory Chaitin, Paradoxes of randomness, Complexity 7 (2002), 14–21. Also available as <http://www.umcs.maine.edu/~chaitin/summer.html>

So, true randomness is somewhat elusive. It seems hard to come by except in quantum mechanics. For example, the time at which a radioactive atom decays is believed to be \*really\* random. I'll be pissed off if it turns out that God (or his henchman Satan) is fooling us by simulating quantum mechanics with some cheap pseudorandom number generator!

Similarly, we can define a "pseudorandom sequence" to be one that no \*efficient\* algorithm can guess with a success rate higher than chance would dictate.

Efficiency is a somewhat vague concept. It's popular to define it by saying an algorithm is "efficient" if it runs in "polynomial time": the time it takes to run is bounded by some polynomial function of the size of the input data. If the polynomial is

$$p(n) = 1000000000000000 n^{1000000000000000} + 1000000000000000$$

most people wouldn't consider the algorithm efficient, but this definition is good for proving theorems, and usually in practice the polynomial turns out to be more reasonable.

A "pseudorandom number generator" needs to be defined carefully if we want to find efficient algorithms that do this job. After all, no efficient algorithm can produce a sequence that no efficient algorithm can guess: \*it\* can always guess what it's going to do!

So, the basic idea is that a pseudorandom number generator should be an efficient algorithm that maps short truly random strings to long pseudorandom strings: we give it a short random "seed" and it cranks

## This Week's Finds in Mathematical Physics (Week 226)

out a lot of digits that no efficient algorithm can guess with a success rate higher than chance would dictate.

If you want a more precise definition and a bunch of theorems, try these:

15) Oded Goldreich, papers and lecture notes on pseudorandomness available at [http://www.wisdom.weizmann.ac.il/~oded/pp\\_pseudo.html](http://www.wisdom.weizmann.ac.il/~oded/pp_pseudo.html)

M. Luby, Pseudorandomness and Cryptographic Applications. Princeton, NJ: Princeton University Press, 1996.

Unfortunately, nobody has proved that pseudorandom number generators exist! So, this whole subject is a bit like axiomatic quantum field theory, or the legendary Ph.D thesis where the student couldn't produce any examples of the mathematical gadgets he was studying. It's a risky business proving results about things that might not exist. But in the case of pseudorandom number generators, the subject is too important not to take the chance.

One of the most interesting things about pseudorandom number generators is that they let us mimic probabilistic algorithms with deterministic ones. In fact there are some nice theorems about this. Let me sketch one of them for you.

As I already mentioned, a "probabilistic algorithm" is just a deterministic algorithm that's been equipped with access to a (true) random number generator. Just imagine a computer with the ability to reach out and flip a coin when it wants to. A problem is said to be in "BPP" – "bounded-error probabilistic polynomial time" – if you can find polynomial-time probabilistic algorithms that solve this problem with arbitrarily high chance of success.

It's a fascinating question whether randomness actually helps you compute stuff. I guess most computer scientists think it does. But, it's tricky. For example, consider the problem of deciding whether an integer is prime. Nobody knew how to do this in polynomial time... but then in 1977, Solovay and Strassen showed this problem was in BPP. This is one of the results that got people really excited about probabilistic algorithms.

However, in 2002, Maninda Agrawal, Neeraj Kayal and Nitin Saxena showed that deciding whether numbers are prime is in P! In other words, it can be solved in polynomial time by a plain old deterministic algorithm:

16) Anton Stiglic, Primes is in P little FAQ, [http://crypto.cs.mcgill.ca/~stiglic/PRIMES\\_P\\_FAQ.html](http://crypto.cs.mcgill.ca/~stiglic/PRIMES_P_FAQ.html)

Is  $BPP = P$ ? Nobody knows! But if good enough pseudorandom number generators could be shown to exist, we would have  $BPP = P$ , since then we could use these pseudorandom numbers as a substitute for truly random ones. This is not obvious, but it was proved by Nisan and Wigderson in 1994.

## This Week's Finds in Mathematical Physics (Week 226)

Here's a great review article that discusses their result:

17) Luca Trevisan, Pseudorandomness and combinatorial constructions, available as cs.CC/0601100.

I recommend reading it along with this:

18) Scott Aaronson, Is P versus NP formally independent?, available at <http://www.scottaaronson.com/papers/pnp.pdf>

This is a delightfully funny and mindblowing crash course on logic. It starts with a review of first-order logic and Goedel's theorems, featuring a dialog between a mathematician and the axiom system  $ZF + \text{not}(\text{Con}(ZF))$ . Here  $ZF$  is the popular Zermelo–Fraenkel axioms for set theory and  $\text{not}(\text{Con}(ZF))$  is a statement asserting that  $ZF$  is not consistent. Thanks to Goedel's second incompleteness theorem,  $ZF + \text{not}(\text{Con}(ZF))$  is consistent if  $ZF$  is! The mathematician is naturally puzzled by this state of affairs, but in this dialog, the axiom system explains how it works.

Then Aaronson zips on through Loeb's theorem, which is even weirder than Goedel's first incompleteness theorem. Goedel's first incompleteness theorem says that the statement

This statement is unprovable in  $ZF$ .

is not provable in  $ZF$ , as long as  $ZF$  is consistent. Loeb's theorem says that the statement

This statement is provable in  $ZF$ .

\*is\* provable in  $ZF$ .

Then Aaronson gets to the heart of the subject: a history of the P vs. NP question. This leads up to the amazing 1993 paper of Razborov and Rudich, which I'll now summarize.

The basic point of this paper is that if P is not equal to NP, as most mathematicians expect, then this fact is hard to prove! Or, as Aaronson more dramatically puts it, this conjecture "all but asserts the titanic difficulty of finding its own proof".

Zounds! But let's be a bit more precise. A Boolean circuit is a gizmo built of "and", "or" and "not" gates, without any loops. We can think of this as computing a Boolean function, meaning a function of the form:

$$f: \{0,1\}^n \rightarrow \{0,1\}$$

Razborov and Rudich start by studying a common technique for proving lower bounds on the size of a Boolean circuit that computes a given function. The technique goes like this:

## This Week's Finds in Mathematical Physics (Week 226)

- A) Invent some way of measuring the complexity of a Boolean function.
- B) Show that any Boolean circuit of a certain size can compute only functions of complexity less than some amount.
- C) Show that the function  $f$  has high complexity.

They call this style of proof a "natural" proof.

The P versus NP question can be formulated as a question about the size of Boolean circuits – but Razborov and Rudich show that, under certain assumptions, there is no "natural" proof that P is not equal to NP.

What are these assumptions? They concern the existence of good pseudorandom number generators. However, the existence of these pseudorandom number generators would follow from  $P = NP$ ! So, if  $P = NP$  is true, it has no natural proof.

Aaronson says this is the deepest insight into the P versus NP question so far. I would like to understand it better – I explained it very sketchily because I don't really understand it yet. Aaronson recommends us to these papers for more details:

19) A. A. Razborov, Lower bounds for propositional proofs and independence results in bounded arithmetic, in Proceedings of ICALP 1996, 1996, pp. 48–62. Also available at <http://genesis.mi.ras.ru/~razborov/icalp.ps>

20) R. Raz,  $P \neq NP$ , propositional proof complexity, and resolution lower bounds for the weak pigeonhole principle, in Proceedings of ICM 2002, Vol. III, 2002, pp. 685–693. Also available at <http://www.wisdom.weizmann.ac.il/~ranraz/publication/Pchina.ps>

21) S. Buss, Bounded arithmetic and propositional proof complexity, in Logic of Computation, ed. H. Schwichtenberg, Springer–Verlag, 1997, pp. 67–122. Also available at <http://math.ucsd.edu/~sbuss/ResearchWeb/theoria/index.html>

(Why don't these guys use the arXiv??)

Also, here are some lecture notes on Boolean circuits that might help:

22) Uri Zwick, Boolean circuit complexity, <http://www.math.tau.ac.il/~zwick/scribe-boolean.html>

Aaronson wraps up by musing on the possibility that the P versus NP question independent from strong axiom systems like Zermelo–Fraenkel set theory. It's possible... and it's possible that this is true and unprovable!

So, there is a fascinating relationship between one–way functions, pseudorandom numbers, and incompleteness – but it's a relationship

## This Week's Finds in Mathematical Physics (Week 226)

shrouded in mystery... and perhaps inevitably so. Perhaps it will always remain unknown whether this mystery is inevitable... and perhaps this is inevitable too! And so on.

Here's a question for you experts out there: have people studied Goedel-like self-referential sentences of this form?

The shortest proof of this statement has  $n$  lines.

I hear that people \*have\* studied ones like

The shortest proof that  $0 = 1$  has  $n$  lines.

and they've proved that the shortest proof of \*this\* has to keep getting longer with larger  $n$ , as long as one is working in a sufficiently powerful and consistent axiom system. This is fairly obvious if you know how Goedel's second incompleteness theorem is proved... but it's possible that some interesting, nonobvious lower bounds have been proved. If so, I'd like to know!

---

"Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin." – John von Neumann

"He is the Napoleon of crime, Watson. He is the organizer of half that is evil and of nearly all that is undetected in this great city. He is a genius, a philosopher, an abstract thinker...." – Sherlock Holmes

---

Previous issues of "This Week's Finds" and other expository articles on mathematics and physics, as well as some of my research papers, can be obtained at

<http://math.ucr.edu/home/baez/>

For a table of contents of all the issues of This Week's Finds, try

<http://math.ucr.edu/home/baez/twfcontents.html>

A simple jumping-off point to the old issues is available at

<http://math.ucr.edu/home/baez/twfshort.html>

If you just want the latest issue, go to

<http://math.ucr.edu/home/baez/this.week.html>

