"Algorithmic thinking is likely to cause the most disruptive paradigm shift in the sciences since quantum mechanics."

Could **Your iPod** Be Holding the **Greatest Mystery** in **Modern Science?**

Bernard Chazelle
Princeton University

Futurologists are an amiable bunch, so it is a puzzle why the future has been so cruel to them. From flying cars and self-cleaning houses to that bugaboo of workaholics—the leisure society—the soothsayer's trail is littered with the carcasses of pet predictions turned roadkill.

Gordon Moore need not worry. The co-founder of Intel tried his hand at crystal gazing once—and struck gold. His celebrated "law" makes the outlandish prediction that every 18 months, like clockwork, white-clad technicians will huddle in a silicon wafer clean room and cram twice as many transistors onto a microchip.

Moore's Law has ruled the roost for the last forty years. All the oohs and aahs you hear about the digital revolution are nothing but the squeals humans emit when tickled pink by Moore's Law. From the nice (medical imaging, e-commerce, whole-genome sequencing) to the vital (Xbox, IM, iPod), its rule has been a veritable ticklefest. Moore's Law has been the sizzling cauldron in which savvy cooks have whipped up a dazzling variety of tasty dishes. Without it, the Information Superhighway would be a back alley to Snoozeville; the coolest thing about a computer would still be the blinking lights.

Moore's Law has had a good run but, alas, its days are numbered. By mid-century, a repeal is all but certain. With the heady days of the Incredible Shrinking Chip receding in the past, expect the revolution to grind to a halt; expect pioneers to give way to tinkerers. Bye-bye ticklefest, hello slumber party.

No tears please. Perched atop their towering achievements, computer scientists (the cooks, remember?) will bask in the soothing certainty that their glorious science died at its peak. With a tinge of sadness but not a little pride, they'll chime in unison "There is nothing new to be discovered in computer science now."

If you think you've seen this movie before, you have. A few short years before Einstein turned our world upside down, the great Lord Kelvin bloviated this gem for the ages: "There is nothing new to be discovered in physics now." Not his lordship's finest hour.

Moore's Law has fueled computer science's sizzle and sparkle, but it may have obscured its uncanny resemblance to pre-Einstein physics: healthy and plump—and ripe for a revolution. Computing promises to be the most disruptive scientific paradigm since quantum mechanics. Unfortunately, it is the proverbial riddle wrapped in a mystery inside an enigma. The stakes are high, for our inability to "get" what computing is all about may well play iceberg to the Titanic of modern science.

Brilliant foresight or latest tripe from the Kelvin school of prophecy? Computing is the meeting point of three Big Ideas: universality; duality; self-reference. To this triad, the modern view adds the concept of tractability and the revolutionary algorithmic paradigm. Here's how it works:

Universality

Few would mistake your iPod for an IBM Blue Gene/L—the world's fastest computer. Yet, fundamentally, the two are the same. Why is that? At the heart of your iPod is a written document made of two parts: "*program, data*." The data section stores the songs as long sequences of 0s and 1s. The program section explains in words (again, 0s and 1s) how to read the data and turn it into sound. Add to this mix a smattering of hardware, the *control*, to read the program and follow its instructions, and voilà: you've got yourself an iPod. The beauty of the scheme is that the control need not know a thing about music. In fact, simply by downloading the appropriate program/data document, you can turn your iPod into an earthquake simulator, a word processor, a web browser, or a paperweight. Your dainty little MP3 player is a *universal* computer.

Separating *control* (the hardware) from *program* (the software) was the major insight of Alan Turing—well, besides this little codebreaking thing he did in Bletchley Park that helped

win World War II. The separation was the key to universality. No one had seen anything quite like it before. At least not since the Chinese philosopher opined: "Give a man a fish and you feed him for a day. Teach a man to fish and you feed him for a lifetime."

In Confucius's hands, the specialized view of

fishing = river + fisherman

finds itself replaced by a universal one:

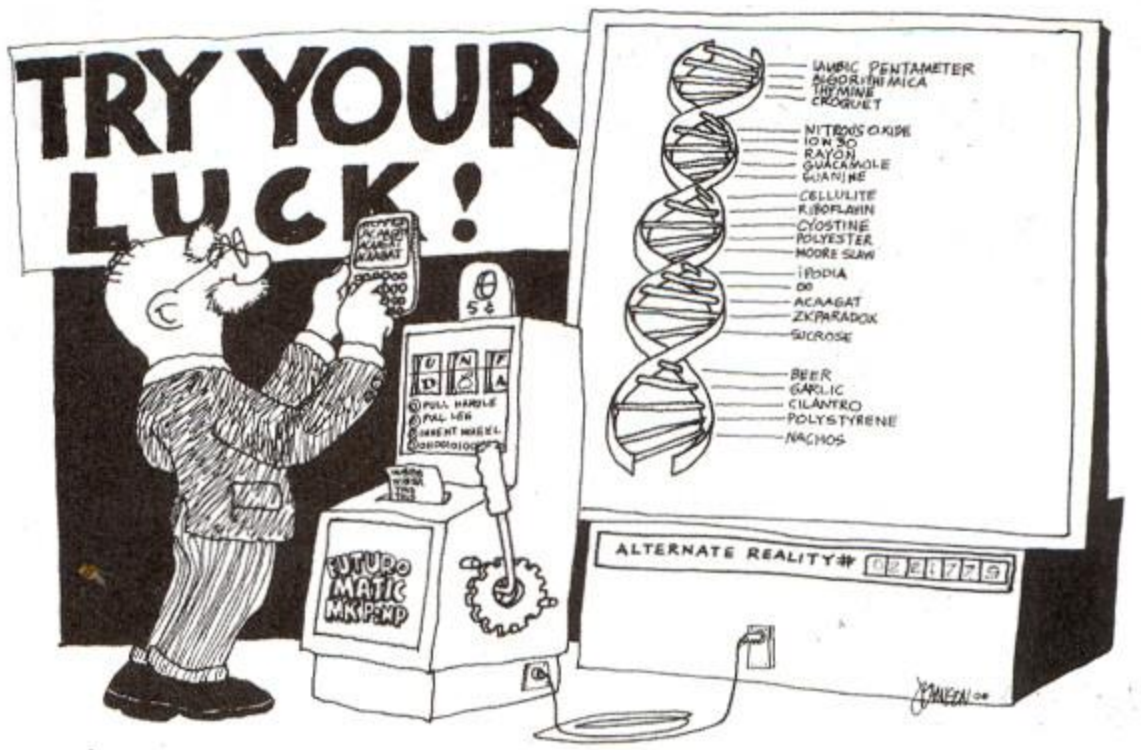
fishing = river + fishing manual + you.

There you have it,

computing = data + program + control.

The control part of your iPod is a marvel of electronics, but the shocker is that it need not be so: universal computers can be built with control boxes vastly simpler than a cuckoo clock. For all purposes,

computing = data + program.



Cartoon by John Johnson

Duality

Consider the iPod document "*Print this, Let 'em eat cake*". Push the start button and watch the words "*Let 'em eat cake*" flash across the screen. Note how the program part of the document, *Print this*, is interpreted as a command—printing is what it wants and printing is what it gets. Contrast this with the data part, *Let 'em eat cake*, which is treated as plain text: no one's eating anything (to Marie-Antoinette's later chagrin). Strings of 0s and 1s are interpreted in one of two ways: as form (data) or as content (program). Tapping into the comic, artistic, and academic potential of this duality, great minds went to work: Abbott and Costello ("Who's on First?"), Magritte ("Ceci n'est pas une pipe"), and Saussure ("signified vs. signifier"). Staring at the sublime will, of course, send the deeper thinkers among us rushing for the classics—such as Homer Simpson's immortal quip: "Oh Marge, cartoons don't have any deep meaning; they're just stupid drawings that give you a cheap laugh."

Self-Reference

Write the iPod document "*Print this twice, Print this twice*" and press the start button. The screen lights up with the words: "*Print this twice, Print this twice*". Lo and behold, the thing prints itself! Just like a computer virus (remember, I did *not* teach you this). The magic word is "twice." For example, the iPod document "*Print this, Print this*" prints this: "*Print this*"—more Dr. Seuss than self-replication.

The "Big Ideas" were the air that the Gang of Four, Princeton branch, breathed all day—that would be Alonzo Church, Alan Turing, Kurt Gödel, and John von Neumann. Mother Nature, of course, figured it all out a few billion years earlier. Reformat your genome by lining up the two strands of DNA one after the other, so it looks like a regular program-data iPod document (billions of letters long though):

"ACAAGAT…GCCATTG, TGTTCTA …CGGTAAC ".

The base pairings (A,T) and (C,G) ensure that the two strands spell the same word with different letters. So, we lose no genomic information if we translate the "data" part and rewrite the whole document as the duplicated text

"ACAAGAT…GCCATTG, ACAAGAT …GCCATTG ".

This is the biological analog of "*Print this twice, Print this twice*". Life's but a walking shadow, Macbeth warned us. Not quite! Life's but a self-printing iPod. Offended souls will bang on preachily about there being more to human life than the blind pursuit of self-replication—though Hollywood's typical fare would seem to refute that. Existential angst aside, *duality* is the option we have to interpret the word ACAAGAT…GCCATTG either as genes (the "form" encoding our genome) or as proteins (the "content" mediating the DNA replication). *Self-reference* is the duplication embodied in the base pairings. Viewed through the computing lens,

life = duality + self-reference.

In the 1953 *Nature* article that unveiled to the world the structure of DNA, Watson and Crick signed off with this lovely understatement: "*It has not escaped our notice that the specific pairing we have postulated immediately suggests a possible copying mechanism for the genetic material.*" Duality and self-reference embedded in molecules: what sweet music to Turing's ears this must have been! Alas, our war hero was a little distracted at the time, busy as he was enjoying the rewards

Continued on page 30

Continued from page 15

that the British authorities had lavished upon him for saving millions of lives during World War II—generous rewards like a court conviction for homosexuality with a sentence of forced estrogen injections. Almost one year to the day of Watson and Crick's triumph, Alan Turing went home, injected cyanide into an apple, ate it, and died. His mother preferred to believe it was an accident.

Tractability

The genesis of this fourth Big Idea was the ho-hum observation that checking the validity of a math proof tends to be much easier than finding the proof in the first place. But is it really? Amazingly, no one knows.

Welcome to the most important open question in all of computer science! Ever wondered if your iPod's 5000-tune library is rich enough to let you compile a playlist of a thousand songs, no two which have ever been played back-to-back on MTV? Let's hope not, because not even an IBM Blue Gene/L could do the job in less time than has passed since dinosaurs were last seen roaming the earth. To find such a playlist (proof-finding) seems hopelessly hard, even on a computer, but to test whether a tentative playlist fits the bill (proof-checking) is a cinch: simply match all possible pairs against MTV's complete playlist, which is readily accessible on the web.

The twin reality of hard proof-finding and easy proof-checking is hardly an MTV-induced aberration. Computer scientists have catalogued over 1000 problems just like it. Of course, courtesy of Murphy's Law, these "Jurassic-1K" include all of the questions humanity is desperate to answer—in artificial intelligence, computational biology, resource allocation, rational drug design, etc.

OK, so life is tough. But since when has that observation qualified as a Big Idea?

Since 1970, roughly. Just as Einstein rebuilt Newtonian mechanics around the constancy of the speed of light, Cook, Edmonds, Karp, and Levin set out to rebuild computing around the notion of *tractability*. A problem is tractable if it can be solved in time growing polynomially in the input size, which is a fancy way of saying "reasonably fast." None of the Jurassic-1K appear to be tractable. At least those in the know believe they are not—of course, not so long ago, those in the know believed the earth was flat. Sadly, the great promise of computing seems to lie with problems afflicted with exponentialitis: the dreaded ailment that places even small-size problems beyond the reach of any computer.

This much we know: it's genetic. If a single one of the Jurassic-1K is tractable then, wonder of wonders, all of them are. These tough puzzles are nothing but different translations of the same Shakespeare play. Heady stuff! The day your playlist question can be answered in a few hours will be the day public-key cryptography dies, bringing down with it all of e-commerce. That day will see biology conquer its highest peak, protein folding, and mathematicians contemplate early retirement. Indeed, the day the Jurassic-1K are shown to be tractable (P=NP in computer parlance), proof-finding will be revealed to be no more difficult than proof-checking. Andrew Wiles, the conqueror of Fermat's Last Theorem, will be found to deserve no more credit than his referees. (Note that this says nothing about *understanding* the proof.) To be P or not to be P, that is NP's question. It is likely that P=NP would do for science what the discovery of the wheel did for land transportation. Little wonder no one believes it.

To discover the wheel is always nice, but to roll logs in the mud has its charms, too. Likewise, the intractability of proof-finding would have its benefits. When you purchase a book from Amazon, the assurance that your transaction is secure is predicated on more than your endearing naiveté. For one thing, it relies critically on the intractability of factoring a number into primes.

Just as modern physics shatters the platonic view of a reality amenable to noninvasive observation, tractability clobbers classical notions of knowledge, trust, and belief. No less. For a taste of it, consider the great *zero-knowledge* (ZK) paradox: two mutually distrusting parties can convince each other that each one holds a particular piece of information without revealing a thing about it. Picture two filthy-rich businessmen stuck in an elevator. Their immediate goal is (what else?) finding out who's the wealthier. ZK dialogues provide them with the means to do so while revealing zero information about their own worth (material worth, that is—the other kind is already in full view.)

Here is a ZK question for the State Department: can a signatory to the Nuclear Non-Proliferation Treaty demonstrate compliance without revealing any information whatsoever about its nuclear facilities? Just as game theory influenced the thinking of cold war strategists, don't be surprised to see ZK theory become the rage in international relations circles.

Tractability reaches far beyond the racetrack where computing competes for speed. It literally forces us to think differently. The agent of change is the ubiquitous *algorithm*.

The Algorithmic Revolution

An algorithm is an iPod program with a human face. If a computer could wash your hair, its program would look like "0110001100100110..." but the algorithm behind it might read: "*Rinse, lather, repeat*." (Don't try this at home if you're a computer scientist.) An algorithm is a list of instructions that tells the computer what to do. It may loop around and entertain alternatives, as in "*Rinse, lather, repeat if unhappy, dry, go to office, answer question: why didn't you rinse the shampoo off your hair*?" An algorithm is, in essence, a work of literature.

The library's bottom shelves might stack the one-line zingers—algorithmic miniatures that loop through a trivial

algebraic calculation to produce *fractals* (pictures of dazzling beauty and infinite intricacy) or print the transcendentally mysterious digits of π. Algorithmic zingers can do everything. For the rest, we have the sonnets on the middle shelves. With names like FFT, RSA, LLL, AKS, they are short and crisp, and tend to pack more ingenuity per square inch than anything else in the computing world. The top shelves hold the lush, richly textured, multilayered novels.

Give it to them, algorithmic zingers know how to make a scientist swoon. No one who's ever tried to compute the digits of π by hand can remain unmoved at the sight of its decimal expansion flooding a computer screen like lava flowing down a volcano. And that's not even the awesome part. For that, one must turn to the infamous Brazilian butterfly whose evil wing flaps cause typhoons in China. Zingers embody the potential of a local action to unleash colossal forces on a global scale: complexity emerging out of triviality. Cellular automata, chaos theory, dynamical systems, and all that.

For a glimpse of the fiction genre on the top shelves, check out PCP. Suppose that, after popping the genius pill, you wake up one bright morning with a complete proof of the Riemann hypothesis in your head (that's the "Notorious B.I.G." of math rap: the biggest open problem in the field). Few number theorists are likely to listen to your story. That is, until you offer them the PCP (Probabilistically Checkable Proof) deal. You'll write down your proof in an agreed-upon format, and then let a "verifier" pick ten lines at random. On the basis of these ten lines alone, the verifier will decide whether your proof is correct. The shocker: beyond any reasonable doubt, he will be right!

The mind reels. If your proof is fine, then it will pass any test the verifier can throw at it. But, based on only ten lines, how can he know that you've proven the Riemann hypothesis and not a baby cousin like 2+2=4? If your proof is bogus, the intuition does not help much either. Presumably, the agreed-upon format is designed to smear any bug all across the proof. But how will the verifier be sure that you didn't play fast and loose with the formatting rules? The PCP algorithm upends basic notions of evidence and persuasion and accomplishes what is usually philosophy's prerogative: to turn the comprehended into the incomprehensible. Somewhere, Wittgenstein must be smiling.

Moore's Law has put computing on the map: algorithms will now unleash its true potential. Physics, astronomy, and chemistry are all sciences of *formulae*. Chaos theory moved the algorithmic zinger to centerstage. The quantitative sciences of the 21st century (e.g., genomics, neurobiology) will complete the dethronement of the *formula* by placing the *algorithm* at the core of their modus operandi. Algorithmic thinking is likely to cause the most disruptive paradigm shift in the sciences since quantum mechanics. And,yes, you may trust the future to be kind to this prediction. ■