His Just Deserts: A Review of Four Books

Reviewed by Alvy Ray Smith

Alan Turing: The Enigma: The Centenary Edition
Andrew Hodges
Princeton University Press, May 2012
US$24.95, 632 pages
ISBN-13: 0691155647

Alan M. Turing: Centenary Edition
Sara Turing
Cambridge University Press, April 2012
US$31.99, 193 pages
ISBN-13: 978-1107020580

Alan Turing's Electronic Brain: The Struggle to Build the ACE, the World's Fastest Computer
B. Jack Copeland and others
Oxford University Press, May 2012
US$33.97, 592 pages
ISBN-13: 978-0199609154

Turing: Pioneer of the Information Age
Jack Copeland
Oxford University Press, January 2013
US$21.95, 224 pages
ISBN-13: 978-0199639793

A computer is an existential conundrum masked as an appliance. While appliances aren't transcendent, a computer is doubly so. First, it's the most malleable tool ever invented by mankind. It allows us to do many more things than we can possibly envision. Second, it's the most powerful amplifier that the human mind has ever had. It increases our power to do those things to unimaginable levels. Malleability and Amplification are the twin glories of computation.

Alvy Ray Smith is a computer graphics pioneer whose contributions include the co-founding of Pixar. His email address is alvyray@gmail.com.

DOI: http://dx.doi.org/10.1090/noti1155

St. Turing

Alan Mathison Turing was born in London on June 23, 1912—a century ago, which is the point, of course. Turing was a fellow at King's College of Cambridge by age twenty-two, received an Order of the British Empire by age thirty-four, became a Fellow of the Royal Society by age thirty-eight, and was dead of cyanide poisoning by age forty-one, on June 7, 1954. The four books we review are part of the centenary celebration of this remarkable man. Three are new editions of earlier works, and the fourth is brand new.

Turing was a mystery to us first-generation computer science students in the 1960s. Of course they taught us his great idea of computation but he himself remained a cipher. Rising out of the mist that otherwise obscured his person was the persistent rumor that he had committed suicide. Then suddenly, in 1983, Andrew Hodges published his biography, *Alan Turing: The Enigma*, which told it all. It finally brought Turing the man into sharp focus. It explained the mystery: Turing had been classified top secret in the War, and much of the information about him had been impounded for decades by Britain's Official Secrets Act—especially the fact that he had played a significant role in cracking Nazi Germany's Enigma code at Bletchley Park.

Topping that revelation was another. Turing had been openly, even recklessly, gay when homosexuality was still a crime. The suicide rumor was officially true. When he was arrested for "indecent acts" in 1952, he couldn't use the fact that he had saved England to save himself. The indecent acts trumped the Secrets Act. Given the choice of

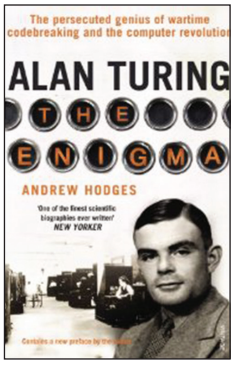
prison or chemical castration, he chose castration. His marathon runner's body fattened from the hormones, and he grew breasts. It was the humiliation, perhaps, that drove him to eat a poisoned apple—in a death scene that almost certainly was lifted straight out of Disney's Snow White.

Biographer Hodges, a King's College theoretical physicist and a member of Britain's gay liberation movement, finally parted the veils of secrecy and embarrassment in Turing's life. He got the full story and told it carefully, intimately, and well. And he brings it up-to-date in the Centenary Edition. It's still the Bible of Turing biography.

The government relaxed its hypocritical anti-gay laws in 1967, and England finally apologized publicly in 2009 for its appalling mistake. But both events came too late to forestall the martyrdom of "St. Turing". The 2012 worldwide celebrations of the centenary of his birth were his vindication—capped off finally by the Queen's pardon on Christmas Eve, 2013. And then there's the fact that his invention is the ubiquitous key to the modern world.

He's Got Algorithm

Intuitively, being careful or systematic about a process means to break it down into a sequence of smaller steps, each of which is simple, unambiguous, and obvious. But what happens when the number of steps in a systematic process gets large, the number of loops through the steps multiplies, and the branches of their possible execution ramify vastly? By asking questions about systematic processes at the turn of the twentieth century, mathematicians started to feel their way toward the notion of computation. They would discover that this new mathematical animal was full of surprises.



Starting in 1900 David Hilbert leveraged his international prestige to focus attention on hard problems. Famously, Hilbert's Second Problem concerned the very foundations of mathematics. His 1928 question did too. He posed it as a question about simple first-order logic.

Hilbert asked if there was a systematic way—an algorithm we would now say—to decide if a statement expressed in the logic is true or not. He did not ask that the algorithm actually generate a derivation of the statement from the axioms of the logic—only that it accurately decide whether one was possible or not. This is curious. If you can decide that a statement is true, why is it important to show the actual derivation of it from the axioms? It's an important distinction.

In the scholarly genealogy of families, for example, it's possible to know that Joseph from the seventeenth century, say, was the direct ancestor of James alive today without formally establishing a father-to-son path, a generation-by-generation descent, between them. If they share the same DNA on the Y chromosome—a straightforward lab test establishes this—then they must be related by a male line. But knowing that a path exists is nothing at all like knowing the actual series of males who passed the particular DNA down the male line—often quite difficult to establish.

Hilbert asked if first-order logic had a trick—like the DNA test—that would decide in a systematic way if a statement was true without actually doing the derivation from the obviously true axioms. This was Hilbert's Entscheidungsproblem, which means decision problem. But while decision problem sounds like a business school topic, *Entscheidungsproblem* suggests a Gotterdämmerung to shake and renew the world. And it did. I'll call it the eProblem since it led to email.

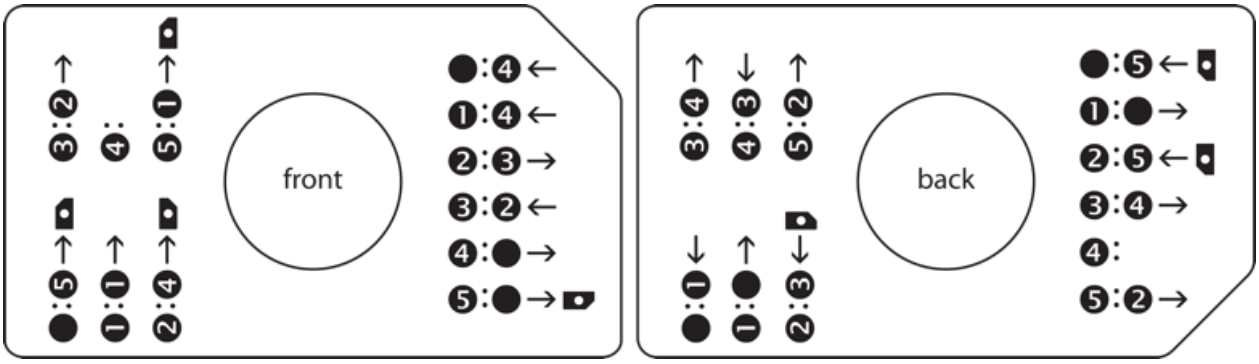
In England in 1934, topologist Max Newman presented the eProblem in a lecture at Cambridge. Student Alan Turing was in attendance. Newman spoke about systematic process—"mechanical process" was the term he actually used. Newman's choice of the words was key. There were no precise words for the concept yet. That was exactly the problem.[1]

The exceedingly literal-minded Turing proceeded to formalize Newman's "mechanical process" with a paper "machine." Then he solved the eProblem with it—in a mighty intellectual leap. Turing used his machine—now called a Turing machine, of course—to show that first-order logic is undecidable. If he had done nothing else—like save Britain or invent computation—this would have put him in the scientific pantheon. He had solved one of the hard problems. But it was his machine—not his solution—that made him famous to the larger world. The modern computer is a direct conceptual descendant of Turing's machine. The path from concept to realization, however, was convoluted, and it was along this path that Turing was to meet perhaps his only failure.

As Turing was solving the eProblem, Alonzo Church at Princeton University was too. In fact, Church beat Turing by several months. By the rules of academia Church had won and the honor would've normally been all his. But Turing's solution technique was strikingly different from Church's, and Newman thought that the mathematical world should know about it.

He urged Church to acknowledge Turing's contribution, and Church did. They both went public, Church slightly before Turing, with printed papers in 1936. This was a big step because it was Turing's intuitive, industrial, even folksy, machine that inspired the birth of the computer, not Church's

[1] *David Anderson, "Historical reflections: Max Newman: Forgotten man of early British computing",* Communications of the Association for Computing Machinery **56** *(May 2013), 29–31.*



abstruse formalization (lambda definability). They were equivalent concepts, of course—Turing proved it so—but Turing's choice of words had profoundly different consequences.

Church wasn't the only other claimant. Emil Post and Stephen Kleene both had equivalent ideas.[2] But Turing's version influenced the modern notion so strongly that, to nonmathematicians, the others pale in comparison. His word computation is the one that stuck. Today we still use the concepts that he introduced. For starters, he gave us programming. That makes him the first programmer. Also, alas, he was the first to write buggy software, as first pointed out by Post.[3]

Martin Davis studied under both Post and Church and wrote influential books explaining Turing to generations of computer scientists.[4] So it's no surprise to find him here as author of the Centenary foreword to Sara Turing's little book in which she attempts, uncomprehendingly, to salvage her son's reputation. Alan's brother John, in the book's unkindest chapter, savages Joan Clarke, Alan's short-term fianceé. Davis's biting analysis of this brotherly betrayal is worth the price of admission alone.

And Turing was the first of another computer tradition. His quirky personality—intense literal-mindedness, honesty to a fault, social awkwardness, and disregard of dress—qualified him as the first geek, too.[5] Newman was afraid that Turing was fast becoming a "confirmed solitary" and told Church so. He asked Church to accept Turing as a graduate student at Princeton, and Church obliged again. Turing would earn his Ph.D. under Church in America.

[2] *Jacques Herbrand and Kurt Gödel are sometimes co-credited with Kleene.*

[3] *Emil Post, "Recursive unsolvability of a problem of Thue",* Journal of Symbolic Logic **12** *(1947), 7.*

[4] *Martin Davis,* The Universal Computer: The Road from Leibniz to Turing, *W. W. Norton & Co., New York, 2000.*

[5] *David Leavitt,* The Man Who Knew Too Much: Alan Turing and the Invention of the Computer, *W. W. Norton & Co., New York, 2006.*

Not a Toy

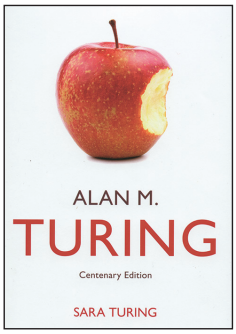
To get a handle on Turing's great idea, consider this business card (above). It has one corner cut off and a round hole in the center. Both the front and the back are inscribed as shown.

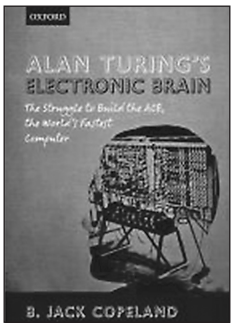
Imagine that there's a paper tape running from left to right behind the card. It's divided into squares, and you can see one square through the hole in the card. The tape is mostly blank, but there are typically one or more squares with symbols on them. I chose the nonblank symbols to be the numerals 1, 2, 3, 4, and 5, but they could just as well be #, !, $, %, and &. The point is that they're distinct marks, without meaning. In particular, they're not numbers. We call them symbols, but they symbolize nothing. Simply replace a 1 with #, a 2 with a !, etc., everywhere in the description of this business card device, and nothing changes—except, of course, the marks.

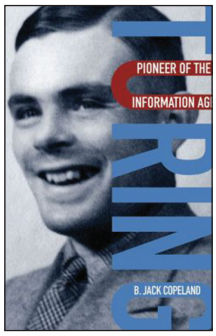
The business card device works like this. Suppose the symbol in the hole is a 5, in front orientation. The rule for 5 is at the lower right in this case. (Pay no attention to rules written sideways.) It says to replace the 5 with a blank then move the card right one square. The little glyph at the right of the rule represents the business card itself. It means that you should rotate the card to match the glyph's orientation. (No glyph means don't rotate.) Now repeat these steps for each new hole position. A rule with no right side means halt.

This isn't an idle game. The business card device is a Turing machine.[6] It's a hardware implementation of Turing's most famous invention. But a modern computer can execute any computation, by simply changing its program. Surely our simple business card device can't execute any computation, can it? Yes, it can. Pixar could compute "Toy Story" with it! They wouldn't want to, however. It's so tediously slow that it might take the lifetime

[6] *Designed 2013 by Alvy Ray Smith, protected by a Creative Commons Attribution-NonCommercial-ShareAlike license. See* `alvyray.com/CreativeCommons/TuringToysdotcom.htm`. *Based on UTM(4, 6) proved universal by Yurii Rogozhin, "Small universal Turing machines",* Theoretical Computer Science ***168*** *(1996), 215–240.*







of the universe—but speed is a separable issue. The point is that this device isn't just any ordinary Turing machine. The business card machine is a universal Turing machine.

Turing's first great idea is that what we mean by a systematic process is embodied exactly in a Turing machine. That idea—the Church-Turing Thesis—is a good one in itself. But Turing's master stroke was to show that there's a single Turing machine that can do what any other Turing machine can. It can perform all systematic processes. It's one machine that can compute anything that's computable. The modern computer is a descendant of this, the universal Turing machine.

Turing did it by encoding the description of an arbitrary Turing machine—an arbitrary algorithm—into a string of symbols. He placed this coded description on the tape of his universal machine. We call that a program today, and programmers call it code. He also similarly placed the data of the arbitrary machine somewhere else on the universal machine's tape. That machine then had enough information to simulate the arbitrary machine on arbitrary data. The simulation is a systematic process, so not surprisingly Turing could design a Turing machine to do it—namely the universal machine. To change what it does, just change the program. So Turing also invented the key notion of the stored-program computer. What we now mean by the single word computer is a universal stored-program machine. We drop it into hardware only to make it go fast.

How many programs can a computer compute? Well, there're so many that you can't count them. It's like asking, how many pieces of music can a piano play? The computer is the most malleable tool ever invented by mankind.

Nexus

Turing proceeded to Princeton for his Ph.D. studies in the late 1930s. His mentor, Newman, soon came for a six-month visit to the neighboring Institute for Advanced Study—sometimes called the Princetitute to distinguish it from the university. John von Neumann—another major player in the story—was already there. Earlier in the decade he had taken a stab at Hilbert's Second with an improvement to Kurt Gödel's as yet unpublished incompleteness result, but Gödel had beat him to it. Hardly missing a beat, von Neumann attempted to recruit Turing to the Princetitute. Astonishingly, Turing rejected this plum offer.[7]

[7] *George Dyson,* Turing's Cathedral: The Origins of the Digital Universe, *Pantheon Books, New York, 2012.*

Turing's and von Neumann's personalities were diametrically opposed, however. Von Neumann was a bon vivant, wore ridiculous party hats, and loved the good salacious limerick.[8] A team of the geek and the bon vivant might not have worked. There was to be no telling, however, because Turing, true to character, struck out on his own. He returned to England, where he was almost immediately recruited into Bletchley Park. It was 1939, and England was frightened for her life.

Bletchley Park

Turing famously helped crack the encryption scheme that the Nazis used for war communications. They employed a devilishly complex encryption machine called Enigma—its actual trade name. It recursively scrambled a text message several layers deep. Descrambling was tedious, superhuman work. The Bletchley Park people built large machines, called Bombes, to aid the humans and increase the speed of decryption. They weren't computers yet, but they were certainly on the path.

Turing needed a partner, not a leader. He was too much the loner. That's where Newman—already his mentor and promoter—would figure again. Like Turing, Newman returned to England from Princeton and joined Bletchley Park.

Turing had led the first-wave attack there. Newman led the second-wave attack, against a newer German encryption machine—nicknamed Tunny. This attack employed giant electronic machines, each called Colossus, the first one built in 1943.[9] These almost-computers were functional years before the almost-computer Eniac in America, on which von Neumann would cut his teeth. But none of these machines was stored-program. They were programmable, but only with hardware cables and toggles.[10]

Turing came up with a mathematical insight, known in Bletchley-speak as Turingismus, that was key to cracking Tunny. Despite his student-teacher relationship with Newman and his own Bletchley Park machine experience with the Bombes, he and Newman didn't team up there—and wouldn't for a while longer—because decoding texts no longer excited Turing. His new interest was encoding voices. The British government sent him back to America on a special mission.

He was to analyze the X System used for secret voice communications between Churchill and Roo-

[8] *Marina von Neumann Whitman,* The Martian's Daughter: A Memoir, *University of Michigan Press, Ann Arbor, 2013.*

[9] *B. Jack Copeland and others,* Colossus: The Secrets of Bletchley Park's Codebreaking Computers, *Oxford University Press, Oxford, 2006.*

[10] *I promote all machine acronyms to real names—Eniac, Ace, Edvac—since that's how we know them and the acronyms are long forgotten.*

sevelt (later Truman). His American counterpart was Claude Shannon. They couldn't talk Enigma secrets but they could talk all they wanted about the possibilities of using computers for an exciting possibility now called artificial intelligence. Computation is about patterns of symbols, not just numbers.

Unknowability

Turing's solution to the eProblem was that there is no decision algorithm. Not surprisingly, considering their common origin with Turing, there is a similar consequence in computation—a certain unknowability—the famous printing problem. You generally cannot know whether a computation will ever print a 1, say. Turing proved there's no algorithm that, given a program and a blank tape, can discover whether the program will eventually print a 1 on its tape. That's a surprising mystery that comes with Malleability.[11]

So a computer is completely determined but not predetermined. It might not be so unsuited to modeling the human brain or mind as many think. Turing certainly thought it was a rich model.

Architecture

It's not farfetched to claim that essentially all computers in use today are descendants of the universal stored-program concept invented by Turing and the architecture for realizing it invented by von Neumann. The von Neumann architecture carries only his name because it appeared alone on the influential report of 1945 which launched it.[12]

Turing had an architecture, too, but it didn't fare so well. He created it for the early computer, Pilot Ace, and its progeny. His machine by all rights should've been the first in the world. Why it wasn't is a tale of bureaucratic bungling and Turing's personal faults, and finally his disillusionment. The Ace book, edited by Copeland, carefully documents the rise and fall of Turing architecture.

The machine that was the first computer was Baby at the University of Manchester, built by Frederic Williams and Tom Kilburn, with first cry in 1948.[13] It used a von Neumann architecture. Pilot Ace, with Turing's architecture, wasn't birthed until 1950. Von Neumann's own machine wasn't first because there was a failure to develop a fast memory device in America. The winning design came from Williams and Kilburn at Manchester—hence Baby. America's first machines eventually used Baby's memory technology too. And Turing, who finally joined Newman at Manchester, wrote programs for Baby and a programming manual for its progeny. Copeland's well-written, and fresh, Turing biography is particularly strong on the Manchester machines—and the Ace machines, of course.

Suicide?

Then there's that suicide—or was it? The official finding was a deliberate act of cyanide poisoning, but the nibbled apple wasn't tested, leaving plenty of wiggle room for alternative theories. The authors reviewed cover the gamut. Hodges is convinced that Turing died by suicide, even though those hormones had worn off by then. But Alan's mother, Sara, never believed it. Neither did Lyn (Irvine) Newman—Max's wife and Alan's dear friend—who wrote the original foreword to Sara's book. Perhaps it was a chemistry experiment gone bad or Alan being sloppy. That's Sara's version. Copeland's is murder. That story of Snow White's poisoned apple is just too precious. He suggests that the British government might have taken Turing out in a fit of McCarthy era insanity.

Amplification

Baby was 10,000 times faster than a human, and Turing's Pilot Ace was even faster. But it was only briefly the "world's fastest computer". The invention of the integrated circuit chip and the announcement in 1965 of "Moore's Law" changed everything. Not the fundamentals, of course—computation would remain Turing's computation, regardless of speed. But Moore's Law told us to expect an order-of-magnitude increase in speed of those computations every five years! Amplification of humans went supernova. It's now reached a quadrillionfold—unimaginable in Turing's time. And it's headed for a quintillionfold by 2025—unimaginable by us even today. That's because we can't see beyond even one order of magnitude, much less three. Those order-of-magnitude barriers are Amplification's unknowability. We just have to get there to see what it means—determined but not predetermined.

[11] *The more famous halting problem wasn't Turing's. Martin Davis,* Computability & Unsolvability*, New York: McGraw-Hill, New York, 1958, p. 70, named and proved it.*

[12] *John von Neumann, "First draft of a report on the EDVAC", Moore School of Electrical Engineering, University of Pennsylvania, 30 June 1945. Unnamed team members included Herman Goldstine, Arthur Burks, Presper Eckert, and John Mauchly.*

[13] *Baby was officially the Small-Scale Experimental Machine.*