DOI:10.1145/2542504

Thomas Haigh

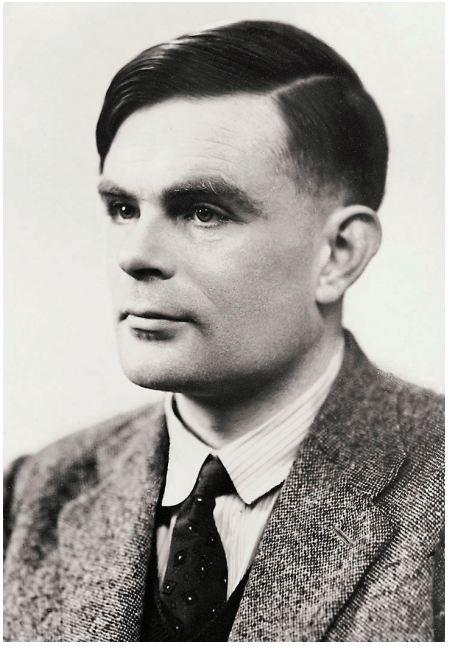
Historical Reflections

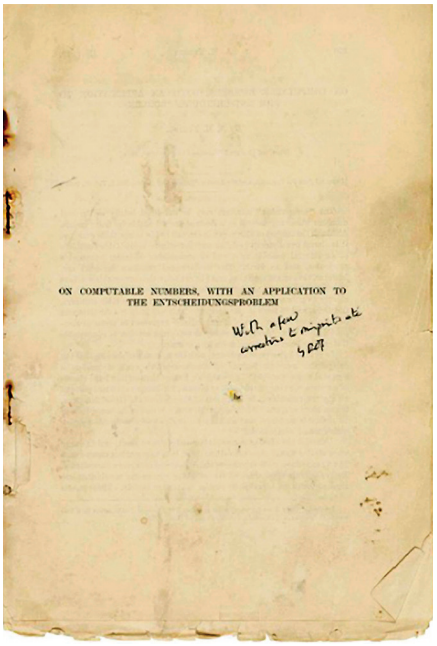
Actually, Turing Did Not Invent the Computer

Separating the origins of computer science and technology.

THE 100TH ANNIVERSARY of the birth of Alan Turing was celebrated in 2012. The computing community threw its biggest ever birthday party. Major events were organized around the world, including conferences or festivals in Princeton, Cambridge, Manchester, and Israel. There was a concert in Seattle and an opera in Finland. Dutch and French researchers built small Turing Machines out of Lego Mindstorms kits. Newspaper and magazine articles by the thousands brought Turing's life story to the public. ACM assembled 33 winners of its A.M. Turing Award to discuss Turing's ideas and their relationship to the future of computing. Various buildings, several roads, and at least one bridge have been named after him.

Dozens of books with Turing's name in the title were published or reissued. Turing was so ubiquitous that even George Dyson's book about John von Neumann was titled *Turing's Cathedral*, becoming the first book on the history of information technology to reach a broad audience since the one about Nazis with punched card machines. Publishers are well aware there is a strong audience for books about Nazis. The public's hunger for books about mathematicians and computer scientists is less acute, making Turing's newfound commercial clout both unlikely and heartening.





Alan Turing (left); the cover page of Turing's paper "On computable numbers, with an application to the Entscheidungsproblem" (right).

Still, as this flood of Turing-related material begins to recede it is time to clean up some of the rather bad smelling historical claims left in our metaphorical basement. Column space is short, so I will focus here on the idea that Turing invented the computer. Very short version: it is wrong.

In case you spent 2012 in a maximum-security prison or meditating in a Tibetan monastery, let me briefly summarize the computer-related high points of Turing's actual career. In 1936, just two years after completing his undergraduate degree, he introduced the concept now called the Turing Machine in a paper called "On computable numbers, with an application to the Entscheidungsproblem." This has since become the main abstract model of computation used by computer scientists. During the Second World War Turing made several vital contributions as part of the British team try-

TURING PHOTO COURTESY OF NATIONAL PORTRAIT GALLERY; PAPER IMAGE COURTESY OF CHRISTIE'S

ing to decipher intercepted German communications, which were encoded using specialized machines and had been thought unbreakable. Immediately after the war Turing designed an electronic computer, the ACE, for the National Physical Laboratory. A series of machines based on the design were eventually built, including one of the first commercial computer models, though Turing departed for the University of Manchester before serious construction began. He worked there with one of the earliest modern computers, but soon turned to more abstract and philosophical questions. Pondering the possibility of what we would now call artificial intelligence, Turing proposed we should judge a computer intelligent if someone could not reliably tell it from a real human after conducting a typed conversation with both. This procedure is now called the "Turing Test." Turing's career came to an abrupt end in 1954 with his death, usually attributed to suicide following various humiliations inflicted by the authorities after a legal conviction for homosexuality.

That is a remarkable career by any measure, with enough tragedy and genius to hook a broader audience and make Turing an unlikely gay icon. I do not have the expertise to evaluate the common claim that Turing's work shortened the war by several years but even a more cautious evaluation of the impact of his wartime accomplishments would make him a mistreated national hero. To celebrate Turing is therefore to celebrate freedom and decency, as well as genius. Let's just make sure we do our cheering in a historically responsible manner.

Retroactively Founding Computer Science

Turing provided a crucial part of the foundation of theoretical computer science. There was no such thing as computer science during the early 1950s. That is to say there were no departments of computer science, no journals, no textbooks, and no community of self-identified computer scientists. An increasing number of university faculty and staff were building their careers around computers, whether in teams creating one-off computers or in campus computer centers serving users from different scientific disciplines. However, these people had backgrounds and appointments in disciplines such as electrical engineering, mathematics, and physics. When they published articles, supervised dissertations, or sought grants they had to be fit within the priorities and cultures of established disciplines. The study of computing always had to be justified as a means, not as an end in itself.

I could fill many columns doing nothing more than skewering ridiculous things written about Turing, many of them by people who ought to know better.

Ambitious computer specialists were not all willing to make that compromise and sought to build a new discipline. It was eventually called computer science in the U.S., though other names were proposed and sometimes adopted. To win respectability in elite research universities the new discipline needed its own body of theory. The minutiae of electronic hardware remained the province of engineering. Applied mathematics and numerical analysis were tied too closely to the computer center tradition of service work in support of physicists and engineers. Thus, the new field needed a body of rigorous theory unique to computation and abstracted from engineering and applied mathematics.

Turing was not, in any literal sense, one of the builders of the new discipline. He was not involved with ACM or other early professional groups, did not found or edit any journal, and did not direct the dissertations of a large cohort of future computer scientists. He never built up a laboratory, set up a degree program, or won a major grant to develop research in the area. His name does not appear as the organizer of any of the early symposia for computing researchers, and by the time of his death his interests had already drifted away from the central concerns of the nascent discipline.

When building a house the foundation goes in first. The foundations of a new discipline are constructed rather later in the process. Turing's 1936 paper was excavated by others from the tradition of mathematical logic in which it was originally embedded and moved underneath the developing new field. In several papers historian Michael S. Mahoney sketched the process by which this body of theory was assembled, using pieces scavenged from formerly separate mathematical and scientific traditions. The creators of computer science drew on earlier work from mathematical logic, formal language theory, coding theory, electrical engineering, and various other fields. Techniques and results from different scientific fields, many of which had formerly been of purely intellectual interest, were now reinterpreted within the emerging framework of computer science.[a] Historians who have looked at Turing's influence on the development of computer science have shown the relevance of his work to actual computers was not widely understood in the 1940s.[1,4,5]

Turing's 1936 paper was one of the most important fragments assembled during the 1950s to build this new intellectual mosaic. While Turing himself did see the conceptual connection he did not make a concerted push to popularize this theoretical model to those interested in computers. However, the usefulness of his work as a model of computation was, by the end of the 1950s, widely appreciated within large parts of the emerging computer science community. Edgar Daylight has suggested that Turing's rise in prominence owed much to the embrace of his work by a small group of theorists, including Saul Gorn, John W. Carr, and Alan J. Perlis, who shared a particular interest in the theory of programming languages.[3] His intellectual prominence has been increasing ever since, a status both reflected in and reinforced by ACM's 1965 decision to name its premier award after him.

a See part three of Mahoney's *Histories of Computing*, cited in the Further Reading section at the end of this column.

So Who Did Invent the Computer?

This question, asked at a party, will cause any responsible historian of computing to blanch and mumble an excuse before scurrying to the safety of the drinks table. The whole way we write and think about the computers of the 1940s is an attempt to avoid having to provide a single answer to that question. Instead we award each early machine, and its main inventor(s), a metaphorical trophy engraved with a phrase such as "first general-purpose automatic electronic digital computer." These trophies adorn the figurative mantelpieces of John Atanasoff, Konrad Zuse, J. Presper Eckert, John Mauchly, Tom Kilburn, Tommy Flowers, Howard Aiken, and Maurice Wilkes. Those who focus on designs, rather than actual functioning machines, can and do make the case for Charles Babbage and John von Neumann. A colleague once joked to me that we should identify and honor the earliest computer never to be claimed as the first computer.

The story behind all those "firsts" goes like this. From the late 1930s to the mid-1940s, a number of automatic computing machines were built. Their inventors often worked in ignorance of each other. Some relied on electromechanical relays for their logic circuits, while others used vacuum tubes. Several machines executed sequences of instructions read one at a time from rolls of paper tape. Thanks in part to a series of legal battles around a patent granted on the ENIAC these machines dominated early discussion of the history of computing and their creation has been well documented.

The "modern" or "stored program" computers from which subsequent computers evolved were defined by two interrelated breakthroughs. On an engineering level, computer projects of the late 1940s succeeded or failed based primarily on their ability to get large, fast memories to work reliably. The first technology proposed, by Eckert who oversaw the engineering of ENIAC at the University of Pennsylvania, was the mercury delay line. Freddy Williams, working on the computer project at Manchester University, was the first to successfully store bits on a cathode ray tube. These were the two dominant high-speed memory technologies until the mid-1950s.

On a conceptual level, the breakthrough was inventing what we could now call a computer architecture able to take advantage of the flexibility of these new memories. Historians agree that the first wave of modern computers under construction around the world during the late 1940s were all inspired by a single conceptual design, an unpublished typescript cryptically titled "First Draft of a Report on the EDVAC." This unfinished document summarized discussions among the team working on a successor to ENIAC. Its title page named only John von Neumann as its author, though the extent to which he personally created the ideas within rather than summarizing the team's progress has been much debated. Turing produced his own ACE design only after reading and being influenced by this document, though his approach diverges in several interesting respects from von Neumann's.

Arguments For Turing

As historians followed this progression of machines and ideas they found few mentions of Turing's theoretical work in the documents produced during the 1940s by the small but growing community of computer creators. Turing is thus barely mentioned in the two main overview histories of computing published during the 1990s: *Computer* by Campbell-Kelly and Aspray, and *A History of Modern Computing* by Ceruzzi.

Much of the overstatement of Turing's role, in newspaper articles or by participants in online discussion, is based on simple misunderstandings. For example, a series of Colossus computers was used by the British for wartime codebreaking work. These were the first electronic digital computers to work properly. People often assume, incorrectly, that Turing must have designed Colossus because he worked at the same secret facility doing closely related work.

I could fill many columns doing nothing more than skewering ridiculous things written about Turing, many of them by people who ought to know better. We will learn more by looking at the best-supported, most careful arguments in favor of the idea that Turing invented the computer. The philosopher Jack Copeland has been one of the most passionate and industrious boosters of Turing's role in recent years, un-

leashing a book on Turing's ACE computer, another on Colossus, a collection of Turing's work, a website full of archival Turing documents, and a series of journal articles. His work continues the influential legacy of logician Martin Davis, whose history of computing *Engines of Logic* presented the universal Turing machine as the crucial advance behind the modern computer.

A painstaking and easily accessible summary of the case for Turing comes is "Alan Turing: Father of the Modern Computer" published by Copeland and Diane Proudfoot in an online journal edited by Copeland.[2] This claims that the "fundamental conception" embodied in the "First Draft Report" came from Turing, and that von Neumann himself "repeatedly emphasized" this. Copeland also believes that "right from the start" Turing was interested in building an actual computer based on the conceptual mechanism described in his 1936 paper. This extends a recent trend, seen for example in George Dyson's book, to write about the teams working to build computers in the late-1940s as if they launched their projects primarily to build practical realizations of Turing's abstract machine.

Copeland is deeply knowledgeable about computing in the 1940s, but as a philosopher approaches the topic from with a different perspective from most historians. While he provides footnotes to support these assertions they are often to interviews or other sources written many years after the events concerned. For example, the claim that Turing was interested in building an actual computer in 1936 is sourced not to any diary entry or letter from the 1930s but to the recollections of one of Turing's former lecturers made long after real computers had been built. Like a good legal brief, his advocacy is rooted in detailed evidence but pushes the reader in one very particular direction without drawing attention to other possible interpretations less favorable to the client's interests.

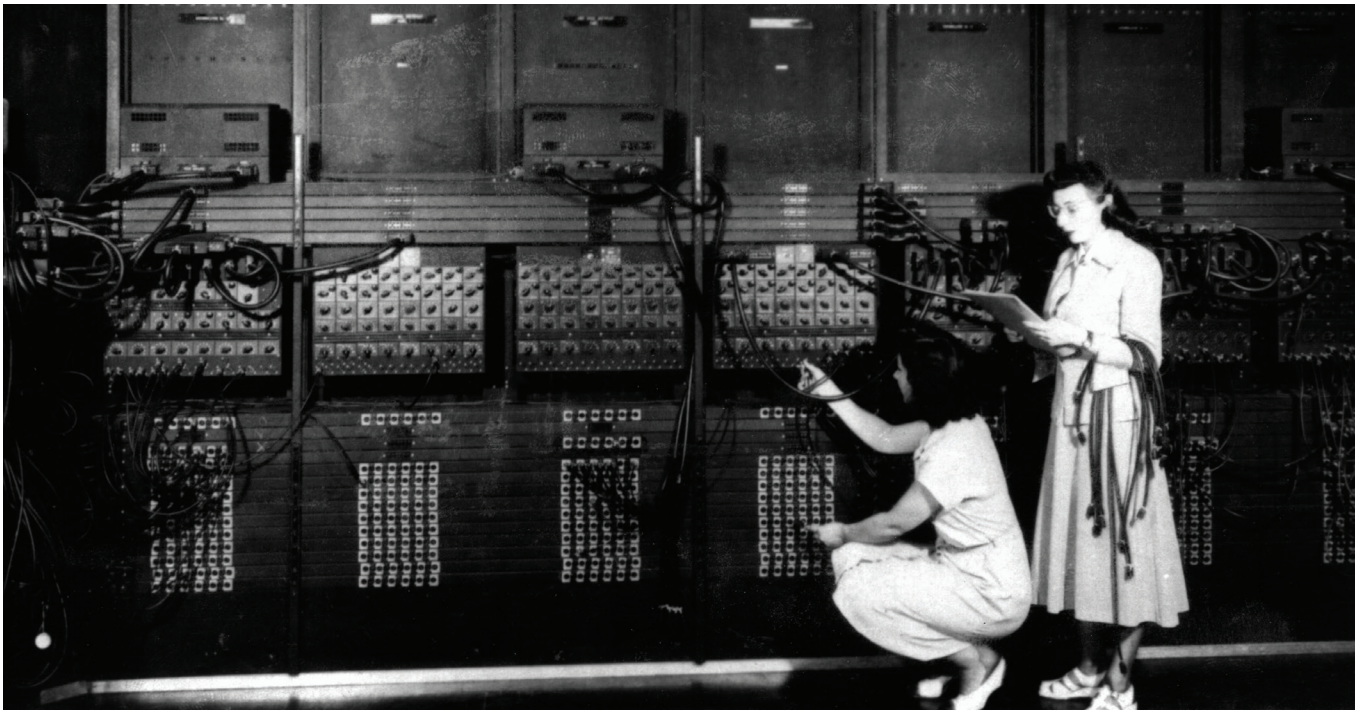
Theory vs. Practice

Arguments of this kind raise fundamental issues about the connection of theory and practice. Are abstract, theoretical insights more fundamental than pragmatic, engineering-based advances? Must theoretical breakthroughs precede and guide practical ones? For a computer scientist, in particular, it is easy to assume that Turing's theoretical work was as centrally important to the computer designers of the 1940s as it later becomes within computer science. There is also something undeniably attractive in the story of a lone genius who anticipates the rest of the world by many years.

Turing's work was not completely unknown in the 1940s. There is, for example, reliable evidence that von Neumann was aware of the now-famous paper and shared Turing's interest in the underlying mathematical questions it addressed.

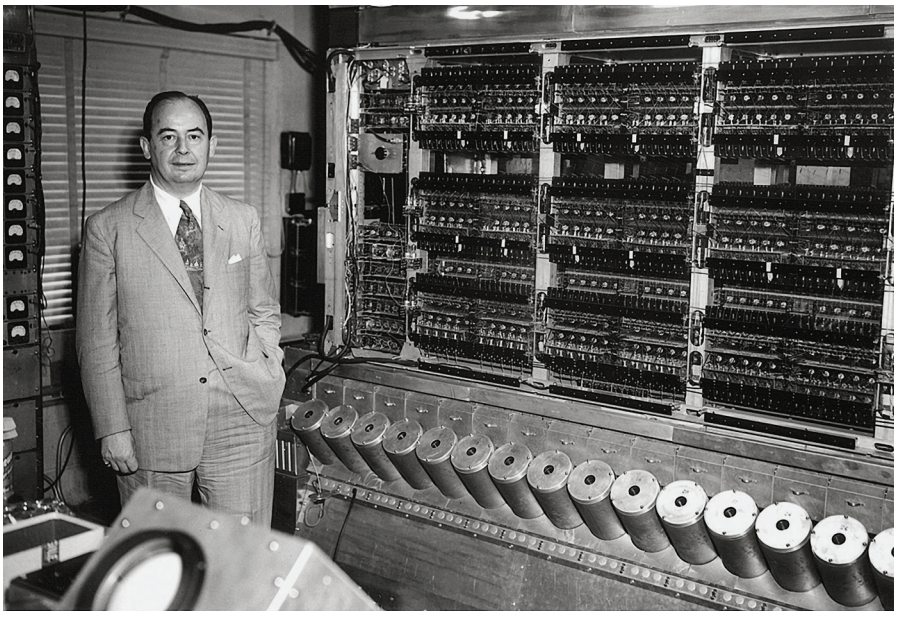
Where one might leap into fantasy is by asserting the cluster of ideas contained in von Neumann's 1945 "First Draft" are merely a restatement, or at most an elaboration, of Turing's earlier work on computability. Judge for yourself, by placing side by side Turing's 1936 "On Computable Numbers..." and "First Draft of a Report on the EDVAC." They are easy to find with Google, though you might want to pour yourself a fortifying beverage first as neither is particularly easy reading.

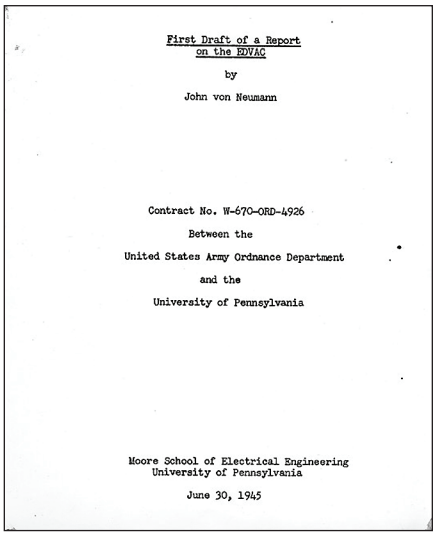
The former is a paper on mathematical logic. It describes a thought experiment, like Schrödinger's famous 1935 description of a trapped cat shifting between life and death in response to the behavior of a single atom. Schrödinger was not trying to advance the state of the art of feline euthanasia. Neither was Turing proposing the construction of a new kind of calculating machine. As the title of his paper suggested, Turing designed his ingenious imaginary machines to address a question about the fundamental limits of mathematical proof. They were structured for



U.S. ARMY PHOTO FROM THE ARCHIVES OF THE ARL TECHNICAL LIBRARY

Two programmers wiring the right side of the ENIAC with a new program.





John von Neumann with the IAS computer circa 1951 (left); cover page of von Neumann's "First Draft of a Report on the EDVAC" (right).

simplicity, and had little in common with the approaches taken by people designing actual machines.

Von Neumann's report said nothing explicitly about mathematical logic. It described the architecture of an actual planned computer and the technologies by which it could be realized, and was written to guide the team that had already won a contract to develop the EDVAC. Von Neumann does abstract away from details of the hardware, both to focus instead on what we would now call "architecture" and because the computer projects under way at the Moore School were still classified in 1945. His letters from that period are full of discussion of engineering details, such as sketches of particular vacuum tube models and their performance characteristics.

The phrase "stored program concept" has sometimes been used to encapsulate the content of the "First Draft" report, but this underplays its actual impact by implying it held just one big idea. In fact it provided a wealth of intertwined ideas and details. In my current work with Mark Priestley and Crispin Rope I have found it useful to separate these into three main areas.[b] The first, the "EDVAC Hardware Paradigm" described an all-electronic binary computer with a much larger memory than anything ever built previously. The second, the "von Neumann Architecture Paradigm," set out the basic structure of the modern computer: special-purpose registers on which all operations were performed and from which data was exchanged with main memory, separation of arithmetic functions from control functions from memory units, only one action performed at a time, and so on. The third, the "Modern Code Paradigm," concerns the nature and capabilities of its instructions. For example, instructions were expressed as through a small vocabulary of operation codes followed by argument or address fields. These were held in the same numbered memory cells as data. While executed by default in a particular sequence, the machine could jump out of sequence and the destination of this jump could be modified as the program ran based on the state of the computation.

Taken together, von Neumann's cluster of ideas guided the construction of computers that were much cheaper, smaller, more reliable, and more flexible than their predecessors. ENIAC, the first general-purpose electronic digital computer, used almost 18,000 vacuum tubes. The more tubes a machine held the more expensive it was to build and, as they eventually burn out, the less reliable. Its immediate successors held 1,000 or 2,000 tubes yet could handle problems of greater logical complexity and were easier to program. This efficiency made possible the construction of computers in cash-strapped Britain following the war, and made computers affordable and useful enough that they were rapidly turned into commercial products and applied to business tasks as well as scientific computations.

The universal Turing Machine has appealed to theorists from the 1950s onward.

According to Copeland, "the fundamental conception of the stored-program universal computer" was Turing's. Von Neumann merely "wrote the first paper explaining how to convert Turing's ideas into electronic form."[c] But what actually would have been different about von Neumann's "First Draft" report if Turing had never written his now famous paper? My answer to that question is: nothing (with the possible exception of the neuron notation he appropriated to describe logic gates, whose creators cited Turing).

Copeland has gone so far as to argue the basic idea of a single machine that could do different jobs when fed

b "Reconsidering the Stored Program Concept," forthcoming in *IEEE Annals of the History of Computing*.

c See http://www.huffingtonpost.com/jack-copeland/what-apple-and-microsoft-_b_3742114.html

PHOTOGRAPH COURTESY OF THE SHELBY WHITE AND LEON LEVY ARCHIVES CENTER, INSTITUTE FOR ADVANCED STUDY (IAS); PAPER IMAGE COURTESY OF THE UNIVERSITY OF PENNSYLVANIA ARCHIVES

different instructions can be traced to Turing. But Charles Babbage had that idea long before, and as mentioned earlier, several computers controlled by sequential instruction tapes had already been constructed with no influence from Turing and were well known to von Neumann before he wrote his report. EDVAC went far beyond this to store a program in addressable internal memory rather than on a sequential instruction tape. To suggest this advance came from Turing is odd, as the machine Turing described had no internal writable memory and took its instructions from a tape. Von Neumann brought a concern with logic and preference for minimal, general-purpose mechanisms to the design of EDVAC but he did not need Turing to teach him that. He was a mathematic genius with a deep pragmatic streak and an astonishing track record of productive collaborations across a huge range of fields.

Turing's 1936 paper lacks many novel and fundamental features found in the "First Draft" such as addressable memory locations. Neither did Turing describe instruction codes followed by arguments, the building blocks of computer programs. The suggestion that the EDVAC design was merely a conversion of Turing's paper implies these features are trivial, and the single important idea in each document is that code and data should be treated interchangeably so programs can modify themselves. Yet while Turing's paper showed one machine could, in modern terms, emulate the functioning of another it never described a machine altering its own instructions. Furthermore, at the very end of the "First Draft" von Neumann expressly forbade EDVAC from overwriting the operation fields in its instructions, even though he relied on modifications to their address fields to accomplish basic operations such as conditional branching. This address modification was a very influential idea in the "First Draft," but was, of course, absent from Turing's paper as his machines did not use addresses. In other words, the capability for unrestricted self-modifying code von Neumann is said to have copied from Turing is something Turing did not describe and von Neumann's design explicitly prohibited.

Computer Science vs. Computing

Our urge to believe the computer projects of the late 1940s were driven by a desire to implement universal Turing machines is part of a broader predisposition to see theoretical computer science driving computing as a whole. If Turing invented computer science, which is itself something of an oversimplification, then surely he must have invented the computer. The computer is, in this view, just a working through of the fundamental theoretical ideas represented by a universal Turing machine in that it is universal and stores data and instructions interchangeably.

This line of thinking blurs the fundamental distinction between building something and modeling it. Copeland shows that as early as 1949 von Neumann alluded to Turing's abstract model of computation as an interesting proof that automata with a certain "minimum level of complexity" could simulate each other's functioning. Yet finding an abstraction useful or provocative as a model of a particular real system does not imply the design of the real system was patterned on the abstraction. An abstraction, ultimately, is useful because of what it leaves out.

To focus on historical computers primarily as embodiments of logical ideas, ignoring the trade-offs their creators made when faced with limited resources and unproven technologies, is to abstract away from the information needed to understand their history and development. Progress in electronic engineering, particularly in memory technologies, created the circumstances in which it began to make sense to think about high-speed digital computers in which instructions were stored electronically. In turn, ideas about the best way to design these machines drove further progress in component technologies and engineering methods.

The universal Turing Machine has appealed to theorists from the 1950s onward precisely because it abstracts away from the complexity of real computer architectures and decouples questions of computability from those of design and engineering. This has been enormously useful for computing theorists, both technically and sociologically. Yet, paradoxically, the world seems increasingly eager to locate the origin of the computer in a mathematical abstraction adopted precisely because it hid all the messy issues of architecture and engineering needed to make any real computer function. Hardware and software are interchangeable to the theorist, but not to the historian. C

Further Reading

Aspray, W.
John von Neumann and the Origins of Modern Computing. **MIT Press, 1990. A thorough and careful survey of von Neumann's many contributions to early computing, including his work on the "First Draft of a Report on the EDVAC."**

Copeland, J.
Turing: Pioneer of the Information Age. **Oxford, 2013. A concise summary of Copeland's work on Turing's ideas and their legacy. He has produced related volumes on Turing's planned ACE computer and the wartime Colossus work.**

Hodges, A.
Alan Turing: The Enigma (Centenary Edition). **Princeton University Press, 2012. An updated edition of the monumental biography that originally put Turing on the road to broader fame.**

Lavington S., Ed.
Alan Turing and His Contemporaries: Building the World's First Computers. **British Informatics Society, 2012. A concise and clearly written expert history, honoring Turing's accomplishments and placing them in the context of British computer developments during the 1940s.**

Levy, P.
"The Invention of the Computer." In Serres, M. (Ed.) ***A History of Scientific Thought.*** **Blackwell, 1995. Concise and thoughtful in its summary of key early computers and their relationship to technologies, applications, and Turing.**

Mahoney, M.S., Ed. Haigh, T.
Histories of Computing. **Harvard University Press, 2011. Section three of this book, "The Structures of Computation," is a provocative selection of papers on the origins of theoretical computer science and its relationship to computation and simulation.**

References

1. Akera, A, *Calculating a Natural World*, MIT, 2006.
2. Copeland, B.J. and Proudfoot, D. Alan Turing: Father of the modern computer. *Rutherford Journal* 4, 2011–2012; http://www.rutherfordjournal.org/article040101.html.
3. Daylight, E.G. Towards a historical notion of 'Turing—The father of computer science.' To appear in *History and Philosophy of Logic*; www.dijkstrascry.com/TuringPaper.
4. Mounier-Kuhn, P. Logic and computing in France: A late convergence. *International Symposium on History and Philosophy of Programming*; http://www.computing-conference.ugent.be/file/12.
5. Priestley, M. *A Science of Operations.* Springer, 2010.

Thomas Haigh (thaigh@computer.org) is an associate professor of information studies at the University of Wisconsin, Milwaukee, and chair of the SIGCIS group for historians of computing.