REVIEW OF *THE ANNOTATED TURING*

I really admire the extreme attention to detail in this book. Petzold doesn't gloss over anything: he calls out issues in Turing's paper from small typos to major errors in formulas; he provides historical context not just on how various thinkers contributed to their fields, but on whether and by which specific books/etc they were likely to have encountered and been influenced by each other.

Turing's 1936/1937 paper is (relatively) famous for proving that it's impossible to write a program that takes any arbitrary program as input and determines whether that program ever halts. Though as Petzold explains, this is not exactly how the paper frames the issue—rather, it divides machines (programs, basically) into two groups that Turing calls "circular" and "circle-free" based on their behavior, and shows there cannot be a machine that determines whether any given machine is circular. This is essentially equivalent to the halting problem, but the latter terminology comes from Martin Davis, years later.

Proving the unsolvability of (what would come to be called) the halting problem was just one step in Turing's paper, not the end goal; the end goal was to prove that Hilbert's *Entscheidungsproblem* (decision problem) was unsolvable. Sadly for Turing, Alonzo Church published his own proof (based on his lambda-calculus) that the *Entscheidungsproblem* was unsolvable before Turing's paper made it to publication. Which sounds like an academic's worst nightmare to me, but Turing and Church went on to collaborate and Turing's paper still went down in history, so I guess it could have been worse.

Here's my attempt at an overview of Turing's paper based on my imperfect understanding of the book (I've probably got some of this embarrassingly wrong, and I'm definitely speaking very loosely):

- Hilbert hoped there was a procedure you could follow that would tell you, for any statement, whether that statement was provable or not according to the rules of some formal mathematical system. The search for such a procedure was called the *Entscheidungsproblem*.
- Church and Turing (in their different ways) proved that there is no such procedure. There are some statements which, no matter how long you search for a proof of them, you will not know whether they are unprovable or you merely haven't stumbled across the proof yet.
- How is this different from Gödel's famous incompleteness theorems? IIUC, something like this: Gödel proved that not all *true* statements are *provable* within a system. Church and Turing proved that you can't even identify all the *provable* statements (let alone all the true ones).
- Turing's basic strategy was: First, he defined a notion of a "machine", which was designed to support all the operations necessary to carry out any possible algorithm. Then he showed that you can't make a machine that determines whether a given machine ever reaches certain states (essentially, the halting problem). Finally he showed how a machine and its states can be represented by statements of propositional logic, such that statements representing future states can be logically derived from statements representing the initial configuration. If a decision procedure existed, you could use it to tell whether any given such statement is provable, which would tell you whether the corresponding machine ever reaches the corresponding state. And since machines can implement any algorithm, you could implement this decision procedure as a machine—but that would be a machine which does the very thing that Turing already proved no machine can do. Conclusion: it's impossible for such a decision procedure to exist.

The book also helped me appreciate the profundity of Turing's conclusions. The unsolvability of the halting problem and the *Entscheidungsproblem* may indicate fundamental limits on our ability to make predictions. From a given starting point, there may be no way to know—*even in principle*—

whether some events will ever happen other than to simply wait and see, or to run a full step-by-step simulation:

> When Stephen Wolfram began studying the complex structures that arise from cellular automata, he tried to find ways to predict the outcomes and perhaps make shortcuts through the generations. He could not, for "there can be no way to predict how the system will behave except by going through almost as many steps of computation as the evolution of the system itself . . . For many systems no systematic prediction can be done, so that there is no general way to shortcut their process of evolution . . ."[1]

1. Charles Petzold, *The annotated Turing: a guided tour through Alan Turing's historic paper on computability and the Turing machine* (Indianapolis (Ind.): Wiley publ, 2008), 350, quotes from Wolfram's *A New Kind of Science* p. 739, 741, 750.

Posted 2024-08-26 by Jacob Williams on brokensandals.net. If you have feedback, email me!