SAVANNAH
LAW REVIEW
VOLUME 5 | NUMBER 1

Halting, Intuition, Heuristics, and Action: Alan Turing and the Theoretical Constraints on AI-Lawyering

Jeffrey M. Lipshaw[*]

[*] Professor of Law, Suffolk University Law School. A.B., University of Michigan; J.D., Stanford University. I decided to participate in the *Savannah Law Review*'s symposium, *Rise of the Automatons*, because of the call for papers, which began with a quote attributed to John von Neumann: "You insist that there is something a machine cannot do. If you tell me precisely what it is a machine cannot do, then I can always make a machine which will do just that." Two paragraphs later, the CFP refers to the coming technological Singularity, "the idea that A.I. will improve itself into a Superintelligence that surpasses the capacity of collective human intelligence."

John von Neumann was simply one of the smartest people living in the twentieth century. He died young and never completed his own project of theorizing and building a self-reproducing machine. (*See infra* note 46.) I am, well, just me. If I want to say there are aspects of automaton-lawyering we should consider in constructing legal education curricula, I feel obliged to dig into the technical weeds. If the essay contributes a "go to" source in the law reviews for a technical explanation of the Halting Problem, all the better.

Any errors are mine alone, but I am grateful to William Byers, Professor Emeritus, Mathematics and Statistics, Concordia University (Montreal). Melanie Mitchell, of Portland State University and the Santa Fe Institute, whose award-winning book on complexity I cite frequently, was generous with her time, comments, and criticisms. If there is anyone in the world capable of envisioning a Kearse without constraints, it would be my old friend John Laird, the John L. Tishman Professor of Engineering in the University of Michigan's Artificial Intelligence Laboratory. John treated me to a couple hours of enlightening conversation about minds and machines. Both Melanie and John were emphatic that they thought the Halting Problem has no bearing on artificial intelligence issues. Nevertheless, as I make clear in the essay, like a few in the AI community and many in other disciplines, I have the temerity to think it's ultimately a philosophical issue. And it is an issue on which, if you make an effort sufficiently to understand the science and math, you don't have to be an AI expert to have a view. In any event, if there's any room here for debate, that's enough to go forward with my thought experiment. Thanks also to Barry Schwartz, Mark Bailey, Gabe Teninbaum, Pat Shin, Andy Perlman, Brian Sheppard, and Steve McJohn.

> Roughly speaking those who work in connection with . . . [computers] will be divided into its masters and its servants. . . . The masters are liable to get replaced because as soon as any technique becomes at all stereotyped it becomes possible to devise a system of instruction tables which will enable the electronic computer to do it for [them]. It may happen . . . that the masters will refuse to do this. They may be unwilling to let their jobs be stolen from them in this way. In that case they would surround the whole of their work with mystery and make excuses, couched in well chosen gibberish, whenever any dangerous suggestions were made.
>
> [T]he centre of gravity of the human interest will be driven further and further into philosophical questions of what can in principle be done etc.
>
> - Alan Turing[1]

Introduction

In the late 1970s, when I first joined a law firm, there was a semi-retired partner in the office next to me who would have begun his career in the 1930s. If he needed to create a document, he would write it out on a legal pad. Then he would call his secretary into the office, who would take shorthand as he dictated from what he had written. Then she would type up her notes and give him the draft. He would edit the draft and return it to her. This cycle would continue until he was satisfied.

Fast forward about ten years. During the 1980s, our firm had installed the Wang system for intra-firm communications and word processing. It's hard to imagine this now, but there were no PCs. There was no WYSIWYG. There were no apps. The Wang system ran on a mainframe. Each lawyer and support staff had a terminal. You could ask your assistant to type up a document for optical scanning, but you could also create original documents yourself, and then do the editing. I was still a litigator, and one Friday afternoon our client received a complaint along with an ex-parte TRO and motion for temporary injunction. My office was downtown, but we had a satellite office close to my house in the suburbs. I went there over the weekend and worked at a terminal in the computer room. By Monday morning, without the assistance of anyone else, I had in final form all of the papers I need to respond to the complaint and the TRO. The experience of being able to do it on my own was revelatory.

Fast forward another five years. I had left the firm and was the general counsel of a large division of a multi-national corporation. The corporation was at the forefront of the manufacturing revolution of the late twentieth century, the transition from the mass production of Henry Ford and Alfred Sloan to the lean production systems developed by Toyota and others in Japan. It was a world of Total Quality Management, Six Sigma, statistical production control, and cycle time reduction. Figuring out how that applied to lawyering was a challenge, but

[1] A.M. Turing, *Lecture on the Automatic Computing Engine* [hereinafter Turing, *Lecture*], *in* The Essential Turing 392 (B. Jack Copeland ed., 2004).

we were part of the team, and game to give it a try. One of the things that was apparent to me was that having people do mechanical tasks rather than thinking and judgmental tasks was wasteful. My secretary had retired, and I made it clear that I didn't need a traditional secretary. What I wanted was a professional to whom I could delegate problems to be solved rather than tasks to be performed.

Fast forward again (and finally) to the summer of 2017. I was at a gym in northern Michigan, chatting with another exercise junkie of about my vintage. It turns out he had a small firm of his own in a mid-Michigan city, specializing in family law. We were talking about professional life toward the end of one's career (something, given my move to academia after long practice, lawyers like to explore with me). He said, "It's hard to wind down, because I really can't hand my clients off to another lawyer. In what I do, they want to deal with me, not with somebody else in the firm." I responded, "Interesting. One of the big topics nowadays is the extent to which technology and artificial intelligence will replace human lawyering. I suspect a client that won't even deal with one of your partners is unlikely to take advice from Rosie the Robot."

I recoil from the extremes of the debate about lawyers and artificial intelligence. As for the AI true believers, pardon my cynicism, but sometimes I think that worries about the falling sky come from observers, like law professors, who have only had a worm's-eye view of practice—the first couple years in a big firm where the primary task is document review in litigation discovery or transactional due diligence. If that's what you think law practice is, then you may be right in seeing technology as eliminating what you do. I am hardly a technology Luddite. While I don't tweet, I have little patience for unconnected, paperbound colleagues who need to kill trees to communicate. But I am underwhelmed by those futurists of the legal profession who have drunk the artificial intelligence Kool-Aid®, and are proclaiming the end of the profession.

To be fair to the true believers, however, and to respond to defenders of human mental exceptionalism, it may be the end of the profession "as we (document reviewers or standard contract preparers) know it." Richard and Daniel Susskind get it right in the concluding chapters of *The Future of the Professions.*[2] They coin the term "AI fallacy" to describe the belief that, in order to outperform a human, a machine must replicate human thinking.[3] Their point is a valid one and underscores an issue that goes back to Alan Turing's imitation game: if the machine can perform the task, who cares whether it is "thinking like a human" (much less "thinking like a lawyer")? The example par excellence is the chess match in which IBM's Deep Blue defeated Garry Kasparov. Deep Blue won, but not by playing like a human.[4] More recently, the success of AlphaGo, playing the game Go, in which the number of possibilities far exceeds chess and thus challenges the ability of human or machine to play without intuition or

[2] RICHARD SUSSKIND & DANIEL SUSSKIND, THE FUTURE OF THE PROFESSIONS: HOW TECHNOLOGY WILL TRANSFORM THE WORK OF HUMAN EXPERTS (2015).

[3] *Id.* at 45.

[4] *Id.* at 276.

heuristics, is likely another example of a machine outperforming a human, yet not thinking like one.[5]

Consider the evidence of advances in AI lawyering. For example, in July 2017, a company called Bootstrap Legal announced the coming of a Rosie the Robot, at least for one area of sophisticated law practice: "online software that uses artificial intelligence to automate the drafting of legal paperwork for real estate investors raising capital for projects in the range of $2 million or less."[6] The program requests answers to a series of questions about the project, and within two days, delivers "attorney-grade" documents like the private placement memorandum to a lawyer for review.[7] JPMorgan Chase uses a program called COIN ("Contract Intelligence") to interpret commercial loan agreements, replacing 360,000 lawyer-hours per year to a matter of seconds of processing.[8] Hence, denial isn't just a river.

This is a reflection in three parts about the relationship of lawyering and artificial intelligence. My goal is a better understanding of the theoretical constraints of the latter. Part I is an assessment of one particular and crucially important aspect in the theory of digital computation—determining if the program being run will reach a conclusion. This is known as the "Halting Problem." One question at the far reaches of "what in principle can be done" is whether any computing machine presently conceivable could *always*, *on its own*, for *every* possible program, determine whether the program will ultimately generate an answer. The essence of the Halting Problem is that the answer to that specific question is "no." A human might well program the machine to decide a question short of a final answer being generated. But the machine itself would not *always* be able to decide whether it had thought enough and it was time to fish or cut bait.

Part II is a philosophical reflection on what it means to decide something as opposed merely to think about it. Even if we have the ability to think as logically and formally as a computing machine, we seem to have the ability to decide and then to act. My thesis is that we as humans, unlike digital computers, seem *always* in the case of *every* problem to be able to stop thinking and start doing, even if we don't know whether the thinking is or will ever be complete.

---

[5] Cade Metz, *In Two Moves, AlphaGo and Lee Sedol Redefined the Future*, Wired (Mar. 16, 2016), https://www.wired.com/2016/03/two-moves-alphago-lee-sedol-redefined-future/. My earliest introduction to learning machines came at a Saturday morning youth program at the Cranbrook Science Museum in Bloomfield Hills, Michigan in the 1960s. At one session, we learned how to build a computer that we could teach to play and win the game of Nim, using only matchboxes and beans. *See* Martin Gardner, *Mathematical Games*, 206 Scientific American 138, 138 (Mar. 1962).

[6] Robert Ambrogi, *Startup Says It's First Robo-Lawyer for Real Estate Investing*, Law Sites (July 25, 2017), https://www.lawsitesblog.com/2017/07/startup-says-first-robo-lawyer-real-estate-investing.html.

[7] *Id.*

[8] Hugh Son, *JPMorgan Software Does in Seconds What Took Lawyers 360,000 Hours*, Bloomberg Markets (Feb. 27, 2017), https://www.bloomberg.com/news/articles/2017-02-28/jpmorgan-marshals-an-army-of-developers-to-automate-high-finance.

Part III is an assessment of what a law school of the future ought to look like, given this moderate view of the interaction between thinking machines and deciding humans.

I need to issue a word of caution. We all need to be aware that this is largely an exercise of pure reason, not an empirical exercise.[9] In other words, we are thinking about the future, and about theoretical rather than practical limits. We've set in motion the spinning pinwheels of our own minds. If they were to stop and spit out the conclusive answer, the game would be over. Is an answer possible? Will the pinwheel ever stop spinning? Well, there's the problem in a nutshell.

I. The Halting Problem for Lawyer-Automatons

A. The Spinning Pinwheel

Anybody who has worked on a computer (particularly a MacBook) knows what it is like to encounter the spinning pinwheel.[10] The processor is doing something, but it hasn't finished. It's thinking. Or it's stuck. How long will it be before it's done? I find that to be an apt metaphor for my own thinking from time to time. I can spin for a long time on really sticky issues. Sometimes it feels like an infinite loop. The one thing I don't know is when I'm going to stop spinning and come up with the answer. It's possible that either the machine or the human pinwheel has to stop spinning and spit out an answer by a time certain because circumstances dictate it. We have to decide whether to go Route A or Route B by the close of business on December 31. As a human, I can stop my mental processing and flip a coin. I can program the machine to stop and flip a coin.

My thought experiment here is to posit the existence of the most sophisticated lawyer-automaton, named Kearse,[11] one that appears by all accounts to replicate thinking like a lawyer in every respect. So as not to fall into Susskind's AI fallacy, I will accept that Kearse is capable of doing anything a digital computer is theoretically capable of doing: that is, given a finite set of axioms and a finite set of algorithms, it can compute anything that is computable. Is Kearse capable of

[9] For a very recent empirical rather than theoretical assessment of the capabilities of lawyer-automatons, see Frank S. Levy & Dana Remus, *Can Robots Be Lawyers? Computers, Lawyers, and the Practice of Law*, 30 GEO. J. LEGAL ETHICS 501 (2017). For an overview of the kinds of legal tasks presently amenable to automation under the current state of the art, see Harry Surden, *Machine Learning and Law*, 89 WASH. L. REV. 87 (2014).

[10] I have borrowed the spinning pinwheel image from Aatish Bhatia, *The Questions That Computers Can Never Answer*, WIRED (Feb. 5, 2014), https://www.wired.com/2014/02/halting-problem.

[11] When I began writing out this sentence, I was thinking about Ronald Dworkin's Hercules and going to call my automaton something like "Holmes" or maybe "Posner." But why always white males? Hence, Kearse, in honor of Judge Amalya Kearse, with whom I interviewed when she was a partner at Hughes, Hubbard & Reed in 1977 and was one of the nicest and smartest lawyers I ever met. If anyone were ever *not* an automaton, it would have been Judge Kearse. Judge Kearse noted to me that her name is often mispronounced as a homonym of "curse;" she usually explains that "Kearse rhymes with fierce." Letter from Hon. Amalya Kearse to Jeffrey M. Lipshaw (Aug. 10, 2017) (on file with the author).

deciding itself, for *every* program and *every* situation it might ever face, that the program it was running would, on one hand, ultimately generate an answer, or, on the other find itself in an infinite loop with no answer forthcoming?

B. Turing Machines

To answer the question, we need to go back to some history and basics about machine thinking.[12] Today's digitally connected world has its roots in an exercise in abstract mathematics. In the early 1900s, the mathematician David Hilbert famously set forth the view that the goal of all mathematics was to show that it could be expressed "in the form of a complete, consistent, decidable formal system—a system expressing 'the whole thought content of mathematics in a uniform way.'"[13] In 1900, at the International Congress of Mathematicians in Paris, Hilbert set out a list of the twenty-three most important problems in mathematics yet to be solved.[14]

Three of the problems were as follows:

- Can every mathematical statement be proved formally from a given set of axioms? This is the issue of completeness.
- Can only true statements be proved? This is the issue of consistency. For example, if we could prove that sometimes ten plus ten equals twenty and sometimes it equals thirty, the mathematical system would have a problem.
- Is there always a definite procedure (i.e., one that follows a series of logical rules of inference) that can determine in a finite amount of time whether a mathematical statement is true or false? Take the following assertion: there are no three positive integers, x, y, and z, for which $x^n + y^n = z^n$ for any integer n greater than 2. Is there a set of logical steps one could always perform to prove that these formulae would always be true?[15] This issue is known in German as the *Entscheidungsproblem*, or, in English, the decision problem.[16]

As to the first two problems, the mathematician Kurt Gödel published a paper in 1930 demonstrating that if a formal mathematical system is consistent, it cannot

[12] I use "machine thinking" metaphorically. Turing thought that the entire issue whether machines "thought" in the way humans did was a waste of time. That is why he proposed the imitation game. Alan M. Turing, *Computing Machinery and Intelligence*, 59 MIND 433, 441 (1950) [hereinafter Turing, *Intelligence*], *in* THE ESSENTIAL TURING, *supra* note 1, at 441–64.

[13] B. Jack Copeland, *Computable Numbers: A Guide*, [hereinafter Copeland, *Computable Numbers*] *in* THE ESSENTIAL TURING, *supra* note 1, at 46.

[14] MELANIE MITCHELL, COMPLEXITY: A GUIDED TOUR 57–58 (2009).

[15] This is "Fermat's Last Theorem," which the French mathematician Pierre de Fermat announced cryptically in the seventeenth century, and which no mathematician was able to prove until Andrew Wiles did in the 1990s. *See* SIMON SINGH, FERMAT'S ENIGMA: THE EPIC QUEST TO SOLVE THE WORLD'S GREATEST MATHEMATICAL PROBLEM, AT 67–69 (1998).

[16] *See id.* at 138–41.

also be complete.[17] In other words, the first statement and the second statement cannot be true at the same time.[18]

For purposes of understanding the spinning pinwheel on our computer or in Kearse's brain, it is the third statement we need to explore. The law reviews abound with attempts to analogize Gödel's insights about the limits of certain delineated formal mathematical systems to legal thinking, and much of the literature is, at best, a stretch.[19] But when we talk about Kearse we are talking about a formal, albeit mechanized, system, so it seems to me that invoking mathematical proofs related to those systems is fair game. Hence, what follows is a primer on Turing machines and the Halting Problem. Thereafter, we'll look at what this has to do with Kearse's capabilities.[20]

The historical serendipity here is that Turing set out to resolve an abstract mathematical problem, and in the process devised a mechanism that could be replicated in electronic circuits. He wanted to resolve the *Entscheidungsproblem*. If he could show that even one mathematical statement was formally undecidable in a set of discrete steps with a finite number of axioms, then he would have answered Hilbert's question in the negative. The mathematical statement he used to demonstrate it was effectively "this number is computable." For a number to be computable, it had to be a real number whose expression as a decimal was calculable by finite means even if the number went on infinitely. Or, alternatively, the number would be computable if a machine using discrete logical steps and a finite number of axioms could write it down.[21]

[17] *Id.*

[18] *Id.*

[19] *See* Mike Townsend, *Implications of Foundational Crises in Mathematics: A Case Study in Interdisciplinary Legal Research*, 71 WASH. L. REV. 51, 116–29 (1996). I am going to do my best to avoid metaphors as between legal systems themselves and formal logical systems.

[20] There are a number of references in the law review literature to the Halting Problem, but none I have found that are as granular as I intend to be in explaining precisely how it might apply to a lawyer-automaton. *See, e.g.*, Joshua A. Kroll et al., *Accountable Algorithms*, 165 U. PA. L. REV. 633, 652 (2017).

[21] A.M. Turing, *On Computable Numbers, With an Application to the Entscheidungsproblem* (1936) [hereinafter Turing, *On Computable Numbers*], *in* THE ESSENTIAL TURING, *supra* note 1, at 59.

Not all real numbers are computable. The set of all real numbers is continuous, and a single real number is best described as a "slice" or "cut" that divides all the higher numbers from all the lower numbers. Bernard Linsky, *Logical Constructions*, STAN. ENCYCLOPEDIA PHIL., https://plato. stanford.edu/archives/win2016/entries/logical-construction; Erich Reck, *Dedekind's Contributions to the Foundations of Mathematics*, STAN. ENCYCLOPEDIA PHIL., https://plato. stanford.edu/archives/ win2016/entries/dedekind-foundations/. The real numbers divide into rational and non-rational numbers. "Rational numbers" are those which can be expressed as ratios of natural numbers. "Natural numbers" are the counting numbers like 1 or 2. "Irrational numbers" are those which are not rational. Examples are the square root of 2, pi, and *e*, the base number for a natural logarithm. A number, whether rational or irrational, is computable in the sense that some sequence of discrete steps can continue producing it to more decimal places. One headache-inducing upshot of this is that however infinitely large the set of *computable* numbers (those that can be calculated using a fixed number of

To introduce even greater rigor to the concept of computability, Turing imagined just such a computing machine (now referred as a "Turing machine," even though it is a thought experiment, not a physical machine). A Turing machine consists of a scanner that contains a finite set of "states." States (or as Turing called them, "m-configurations") are discrete sets of instructions for the scanner. An imaginary and infinitely long tape runs past the scanner. The tape is divided into squares. Each square of the tape may contain a symbol, usually either 0 or 1, or may be blank. The scanner only "reads" one square at a time. The machine works by being in a particular "state," "seeing" what symbol is or is not in the square, and then acting on the state instructions that tell the machine what to do based on what it "sees" in the square. The scanner has two possible physical operations. It can print or erase symbols in the scanned square, and it can move right or left to a different square. The machine can also remember what it has already printed or erased on the tape.[22]

A complete and discrete instruction for the machine would include the following four elements: (1) its current state; (2) the symbol (or lack of one) in the scanned square; (3) the operations (printing, erasing, or moving) to be performed in that state, given the symbol appearing in the scanned square; and (4) the state to which the machine changes after completion of the operation.[23] Moreover, the machine is automatic in the sense that all it can do is act on the information it has on the tape and the instructions within any given state.[24]

The reason the concept of the Turing machine was useful theoretically was because its combination of symbols on the tape and instructions within the scanner could compute any computable number. The point of all this was to create a definite, or algorithmic, procedure of computation to satisfy Hilbert's articulation of the *Entscheidungsproblem.*[25] To distinguish between a computable number and one that was not computable, Turing distinguished between a machine that was "circular" and one that was "circle-free." That is, a circular machine goes into a loop. It reaches a configuration (i.e., a combination of tape symbol and state) from which there is no next move, or an inability to write any more symbols. A sequence is computable if it can be computed by a circle-free machine. In other words, the machine continues the computation and does not loop.[26]

Melanie Mitchell has a nice example of how a Turing machine would determine if a number (represented on the tape by a sequence of 1s in the squares bounded on each side by a 0) is odd or even. Assume this sequence on the tape:

axioms) may be, it still doesn't include infinitely more *real* numbers that sit between the computable numbers. This was what Georg Cantor proved by way of his diagonal method, referred to *infra* note 37. *See* ROGER PENROSE, THE EMPEROR'S NEW MIND 108–13 (1999).

[22] Turing, *On Computable Numbers*, *supra* note 21, at 59–60; MITCHELL, *supra* note 14, at 61–62.

[23] MITCHELL, *supra* note 14, at 63.

[24] *See generally* Turing, *On Computable Numbers*, *supra* note 21, at 61–63; MITCHELL, *supra* note 14, at 62–65.

[25] *See infra* note 43.

[26] Turing, *On Computable Numbers*, *supra* note 21, at 60–61.

01110. It represents the number 3, which is odd. The machine's job is to read the sequence, erase it, print 1 if it is odd or 0 if it is even, and then halt. The machine would have four states: Start, Odd, Even, and Halt. The combination of inputs from the tape (i.e., the 01110) and the instructions in the machine are the algorithm that carries out the computation. If the machine is in the Start state and reads 0 on the tape, its instruction is to erase the 0, leaving the square blank, move one square to the right, and change to the Even state. If it reads a 0 in the next square, it prints a 0 (representing Even) and changes to the Halt state, i.e., stops. But, in our case, it finds a 1. Its instruction, if finding a 1 while in the Even state, is to erase it, move one square to the right and change to the Odd state. If it were to find a 0 while in the Odd state, it would erase the 0, print 1, and change to the Halt state. In short, the machine has a complete set of instructions that cause it to read and erase until it finds a 0. If the 0 has been preceded by an odd number of 1s, it will print 1 and halt, telling us the sequence was odd. If the 0 has been preceded by an even number of 1s, it will print 0 and halt, telling us the sequence was even.[27]

The common expression for this is that "machine M [(the scanner with its instructions)] [is] running on input I [(the tape)]."[28] M is the program and I is the input on which the program is running.[29] Note that the "machine" here is a one-trick pony. The only instructions embedded in the scanner are those that would allow it to determine if a number represented by a string of 1s was odd or even.

Turing's next step was to reduce the entire machine M to something expressible as a string of numbers. In modern terminology, this would be the exercise of translating a series of steps understandable in English to machine-readable binary code consisting of nothing but 0s and 1s. As discussed above, any computation algorithm would be expressible as a table in which each line of the table showed the four elements of that step: current state, symbol, operation given the symbol, and new state. It turns out that each line of the table could be reduced by convention to a series of symbols separated by semicolons. Turing called this the machine's "standard description."[30] Each symbol in the standard description (i.e., letters and semicolons) could be replaced by Arabic numbers, called "description numbers."[31] These would always be in the form of a (long) string of integers. Thus, the algorithm for the computation of any computable sequence can be reduced to a description number that has a complete meaning because segments of the number represent, in turn, current states, symbols, operations, and new states. Hence, a (very long) description number represents machine M.

[27] MITCHELL, *supra* note 14, at 62–64. Turing's own examples can be found in Turing, *On Computable Numbers*, *supra* note 21, at 61–66.

[28] MITCHELL, *supra* note 14, at 64.

[29] *Id.* at 63.

[30] Turing, *On Computable Numbers*, *supra* note 21, at 67.

[31] In Turing's paper, for example, the semicolon signifying the separation of steps became a 7. 7 is an Arabic numeral in base ten. Turing, *On Computable Numbers*, *supra* note 21, at 676. In base two or binary notation, 7 would be 111. And, indeed, in Melanie Mitchell's explication of the Turing machine, she uses 111 as the separation code. MITCHELL, *supra* note 14, at 64.

If the description number represented a circle-free machine (i.e., one that would satisfactorily perform the calculation rather than going into a loop), in Turing's terminology, it was a "satisfactory" number. To repeat, the point of this entire exercise was to prove "that there can be no general process for determining whether a given number is satisfactory or not."[32] In other words, there is no way of *always* determining conclusively that a particular algorithm will continue the desired computation to the desired halting point without looping.

The final step in the conceptualization of the machine is its universalization. So far in this example, we have Turing machine M, nothing more than an Odd-Even Calculator program, running on input I, a series of 1s in discrete squares of an infinite tape, bounded on each side by a 0. Could you create a machine, called U, in which both the specific machine M and the tape I are inputs to U? In other words, U would run on M and I. The answer is "yes," and U is referred to as a universal Turing machine.[33] In Turing's thought experiment, U has its own infinite tape. M has been reduced to a description number that gets stored on a segment of the tape. Input I gets stored on another segment of the tape. U would have its own instructions (in its own scanner) that would cause it to look at the first square of I and undertake the instructions embedded in M with respect to the symbol appearing in the square.

Return to Melanie Mitchell's Odd-Even Calculator. All of M's instructions, under which it begins in a Start state, identifies a symbol, performs an operation, and changes its state to Odd, Even, or Halt, have been reduced to a description number (we would today call that "machine code"). U would see the symbol in the I square, refer back to the sequence within the tape containing M, and undertake on the I square whatever M told it to do given that symbol. At each step, U would retain a record of what it had done. The result of U's operation would be to replicate precisely what M had done before: count the number of 1s until it reached 0, print 1 if the result was odd or 0 if the result was even, and then halt. That is, M's instruction to change to the Halt state would cause U to halt as well.[34]

C. Kearse's Halting Problem

As we have noted, Turing's original conception of a "machine" was merely to clarify the mechanics of formal proof. By definition, the machine must undertake the logic of the proof in discrete steps under a finite set of rules and axioms. Turing showed there was at least one mathematical statement—effectively, "this number is computable"—for which it could *not* be proved that the machine was either going to continue the computation to a satisfactory

[32] Turing, *On Computable Numbers*, *supra* note 21, at 68.

[33] *Id.* at 68–69; MITCHELL, *supra* note 14, at 64.

[34] MITCHELL, *supra* note 14, at 64–65. Machine U is, conceptually, the basis for every digital computer ever built. All the programs and all the inputs get translated into binary numbers that run on a central processor. The "tape" on which they get stored is finite, but given the advances in processing power and memory, very, very long!

conclusion or go into a loop. This was his resolution of the Halting Problem.[35] The syllogism for my purposes will be as follows:

(1) The Halting Problem exists for any universal Turing machine.

(2) For the foreseeable future, Kearse will be, or will run on, a universal Turing machine.

(3) Therefore, whatever constraints the Halting Problem puts on a universal Turing machine will exist for Kearse.

Let's turn first to the proof that there is no solution to the Halting Problem. Because there is no solution to the Halting Problem for any universal Turing machine, there will not be a solution for Kearse.

1. The Halting Problem in a Turing Machine

Turing's resolution of the *Entscheidungsproblem*, the Halting Problem, is a proof by contradiction.[36] That is, he assumes that a formal method or program (call it Turing machine H) exists that is capable of determining whether another program will halt. If this assumption about H leads to a logical contradiction, H cannot exist. If H cannot exist, it means there would no definite procedure that would always allow one to tell if another program would ever stop running.

Assume we have a universal Turing machine U that will run all of the various programs and inputs that follow. We need to make a choice about something computable. We want the result of the computation to be 1 if the choice is to be "yes" and 0 if it is to be "no." We use program (i.e., Turing machine) P to compute the answer, given certain inputs. We are going use U to run the inputs

[35] "Halting Problem" is the popular term, and I use it here, but it is a little bit of a misnomer. For that reason, I use it interchangeably with *Entscheidungsproblem*. "Halt" in this context means that the machine continues to calculate without going into an infinite loop. Some calculations, like the Odd-Even calculation described *supra* note 27, very clearly "halt" with a finite answer. Some answers are going to be computable yet infinite, given infinite resources. A computer can be programmed to divide 3 into 9, the result of which is an infinite string of 3s to the right of the decimal. Or it can be programmed to calculate an irrational number like pi or the square root of 2. We presume (intuitively, but can't prove it, which is the point of the Halting Problem) the algorithm will continue to spit out a correct sequence of digits which are the correct answer to more and more decimal places. Assuming infinite resources, the machine won't stop running. It will, however, not go into a loop. In Turing's terms, it is "circle-free."

[36] As Melanie Mitchell observed, this is an exercise that gives even computer science graduate students a headache. MITCHELL, *supra* note 14, at 66–68. Turing's own notation is no longer the standard. Turing, *On Computable Numbers*, *supra* note 21, at 72–74. There are many translations and popular demonstrations of Turing's proof. I have primarily used Mitchell's (MITCHELL, *supra* note 14 at 65–68), Copeland's (Copeland, *Computable Numbers*, *supra* note 13, at 36–40), and Penrose's (PENROSE, *SUPRA* NOTE 21, AT 80–87) to work my way through the key Section 8 of *On Computable Numbers*. I took the idea of drawing a flow chart (of my own design) from Bhatia, *supra* note 10. A clever video presentation is available online. *See* Udi Aharoni, *The Halting Problem*, ZUTOPEDIA, http://www.zutopedia.com/halting_problem.html. I have, however, also worked through Turing's actual reasoning process. That summary is in Appendix A. The footnotes to the description that follows tie to the corresponding sections of Appendix A.

on P. That is, P will choose and print 0 on some inputs and 1 on others, and then halt. Figure 1 shows this schematically.

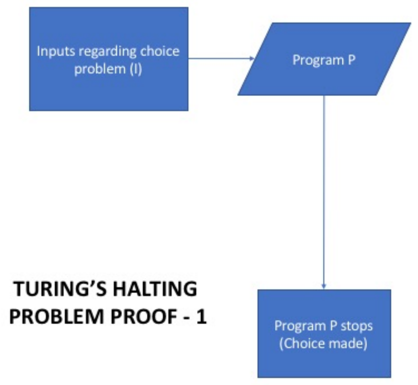
Inputs regarding choice problem (I)
Program P
TURING'S HALTING PROBLEM PROOF - 1
Program P stops (Choice made)

Figure 1.

We have started P running on I. P is chugging along, continuing to run, but it has not yet finished. That is, we will know it has finished when it has printed either a 0 or a 1 and halted. We would like to know if it is going to finish or go into a loop. Is there another program H (another Turing machine) that would look at the process "input I running on program P" and calculate whether P will eventually finish as opposed to going into a loop?[37] We are going to assume that H exists, even if we aren't sure how H would work. H will spit out "yes" (or a 1) if P halts on input I and "no" (or a 0) if it does not. See Figure 2.

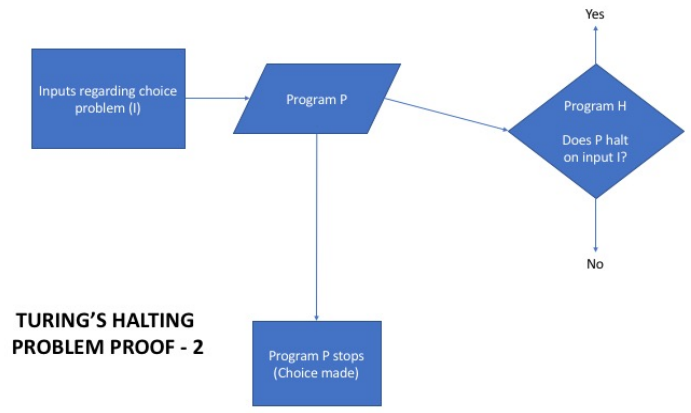
Yes
Inputs regarding choice problem (I)
Program P
Program H
Does P halt on input I?
No
TURING'S HALTING PROBLEM PROOF - 2
Program P stops (Choice made)

Figure 2.

[37] *See* Appendix A ("Turing's First Demonstration") *infra* note 162 and accompanying text.

An example of P might be a program that counts lines of code in a program. If P ran on itself, it would be to determine how many lines of code were in P.[38] We would like to know if P running on P will halt.[39] The way we are going to test this is to create in U a special version of H called H+. H+ will be designed first to run an input of "P running on P itself." Then the "H" portion of H+ will determine if "P running on P" halts. If H+ gets the answer "No, P does not halt when running on P," then H+ itself halts. If it gets the answer "Yes, P halts when running on input P," then H+ will continue running as long as P continues to run. See Figure 3.[40]

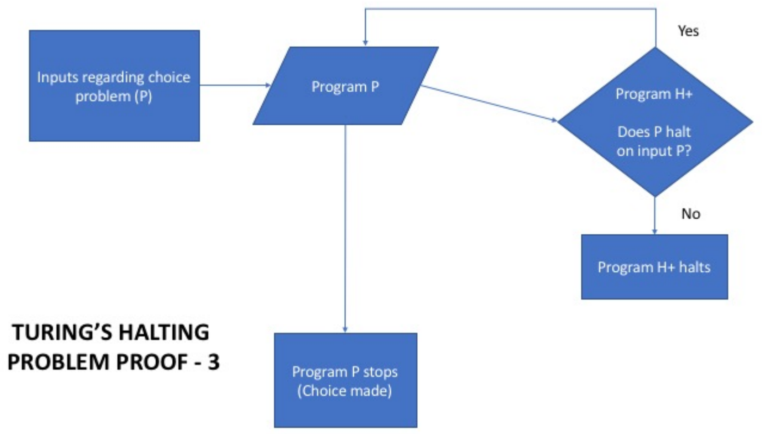
Inputs regarding choice problem (P)
Program P
Yes
Program H+
Does P halt on input P?
No
Program H+ halts
TURING'S HALTING PROBLEM PROOF - 3
Program P stops (Choice made)

Figure 3.

Finally, assume that we let U take all of what H+ is doing (that is, running the input of "P running on P" and then checking if it halts) and run H+ on itself. H+ is the input and H+ is the program. Does H+ halt on input H+? The answer turns out to be a paradox or contradiction. If H+ doesn't halt on input H+, then H+ is done and it halts. Think about what the red boxes in Figure 4 are saying. The top red box is showing that if the answer from H+ is, yes, H+ does halt, then as before, H+ doesn't halt. It goes into a loop.[41] It keeps running on itself as long as it keeps running. The lower red box shows what happens if H+ answers about itself that it doesn't halt. If H+ says it doesn't halt, it halts.

[38] Or a word count program running on itself. *See generally* MITCHELL, *supra* note 14, at 66.

[39] *See* Appendix A ("Turing's Second Demonstration"), *infra* note 163 and accompanying text.

[40] *See id.*, *infra* note 164 and accompanying text.

[41] *See id.*, *infra* note 165 and accompanying text.

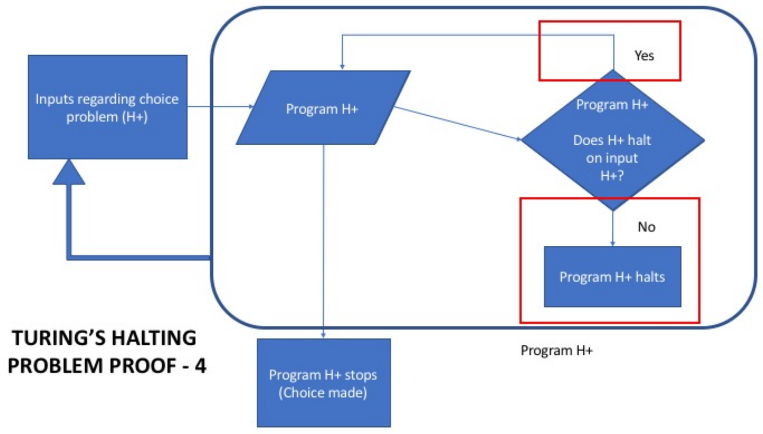
Inputs regarding choice problem (H+)
Program H+
Yes
Program H+
Does H+ halt on input H+?
No
Program H+ halts
Program H+
TURING'S HALTING PROBLEM PROOF - 4
Program H+ stops (Choice made)

Figure 4.

H+ halts when it doesn't halt, and doesn't halt when it halts. That result is problematic. It means that H+ (and H) are not possible.

In plainer language, it demonstrates the answer to the *Entscheidungsproblem*. Not every mathematical statement, such as "this number is computable," is decidable in a formal process of finite axioms, finite rules, and discrete steps. We assumed the existence of a "halting program," H, that will decide formally whether another program, such as one designed to compute numbers, will halt. When we ran the program H on itself, we created a contradiction proving conclusively that our assumption about the existence of program H cannot be true. Once the pinwheel starts spinning for this particular program P, the computer can't tell us by way of another program H within the computer whether the spinning pinwheel means that program P is still working or has gone into a loop.

This exposition of Turing's math is as far as we need to go for our purposes in assessing the capability of universal Turing machines. It was not, however, Turing's final answer to the *Entscheidungsproblem*. In the remainder of his paper, he went on to establish something now known as the Church-Turing thesis, namely, that a universal Turing machine is an idealization of a human being doing the computation, assuming the human is limited to a finite set of conditions or axioms.[42] The point was to generalize the resolution of the *Entscheidungsproblem* to cover Hilbert's question for any computation.[43] Two variants on the thesis are:

1. The universal Turing machine can perform any calculation that any human computer can carry out.
2. Any systematic method can be carried out by the universal Turing machine.[44]

[42] Copeland, *Computable Numbers*, *supra* note 13, at 41.

[43] *See generally* Turing, *On Computable Numbers*, *supra* note 21, at 59, 74–88; Copeland, *Computable Numbers*, *supra* note 13, at 41.

[44] Copeland, *Computable Numbers*, *supra* note 13, at 41.

In the abstract, then, any modern digital computer that runs on stored programs is a universal Turing machine.[45] Kearse, if it were to exist in the foreseeable future, would be a Turing machine and quite capable. But it would also be subject to the mathematical constraints of a Turing machine, namely, the inability to determine for *every* program that it might run whether it would, on one hand, complete that program and generate an answer, or, on the other hand, get stuck in a loop.[46]

2. Intuition, Mistakes, and Heuristics

Not surprisingly, Turing, his contemporaries, and later commentators thought and wrote extensively about the constraints on artificial intelligence, under the view that any machine, including Kearse, would be some version of a universal Turing machine. Turing's own position was interesting. He defended machine "thinking" zealously. Nevertheless, his own work provides the basis for thinking about overcoming the constraints. "[I]f a machine is expected to be infallible, it cannot also be intelligent. There are several mathematical theorems which say almost exactly that. But these theorems say nothing about how much intelligence may be displayed if a machine makes no pretense at infallibility."[47] Turing's position was not that machines could learn to think perfectly or even like humans, but that the positives of machines would outweigh the limitations. Human experts had the luxury of using common sense, learning from mistakes,

[45] *Id.* at 15.

[46] Perhaps if John von Neumann were alive today, he *could* build a computer not constrained by the Halting Problem. He certainly recognized it was an issue. One of von Neumann's last and uncompleted projects before his early death was proving theoretically that one could build a self-reproducing machine. MITCHELL, *supra* note 14, at 123–26. His colleague, Arthur Burks, completed the work after von Neumann's death. JOHN VON NEUMANN, THEORY OF SELF-REPRODUCING AUTOMATA (Arthur Burks ed., 1966) (the full text of the book is available at https://archive.org/details/theoryofselfrepr00vonn_0). Von Neumann appears to have acknowledged that, as long as the automaton was operating like a Turing machine, it would be affected by the Halting Problem.

> Turing proved that there is something for which you cannot construct an automaton; namely, you cannot construct an automaton which can predict in how many steps another automaton which can solve a certain problem will actually solve it. So, you can construct an automaton which can do anything any automaton can do, but you cannot construct an automaton which will predict the behavior of any arbitrary automaton. In other words, you can build an organ which can do anything that can be done, but you cannot build an organ which tells you whether it can be done.

Id. at 51. The Halting Problem is an issue if the automaton needs the capability of self-reference, and von Neumann considered means of resolving it. *Id.* at 125–26.

[47] Turing, *Lecture*, *supra* note 1, at 394. Turing is referring here to the *Entscheidungsproblem* and Gödel's incompleteness theorem. A machine would never be able to answer *every* question, and so for it to be "intelligent," it would have to be able to reach its computational limits, then, like a human, take a guess, with the possibility that the guess would be a mistake. Gualtiero Piccinini, *Alan Turing and the Mathematical Objection*, 13 MINDS & MACHINES 23, 37–38 (2003)

applying short-cut rules of thumb, and making the occasional blunder. Wrote Turing, "Against it I would say that fair play must be given to the machine. . . . No man adds very much to the body of knowledge, why should we expect more of a machine?"[48]

Indeed, as we will see, there are required concessions for machine calculation to resemble more closely human thinking. They involve the introduction of randomness, the possibility of making mistakes, or the ability to say, "I don't know," if it doesn't appear that an answer is computable. In short, making the machines more like humans means making them dumber, or at least more erratic and less formally rational.[49] I am simply assuming here that Kearse can come as close as any conceivably buildable automaton to being human-like and then exploring the theoretical constraints in Turing's own terms.

The famous expression of Turing's thinking on the ultimate capability of machine thinking was *Computing Machinery and Intelligence.*[50] There he proposed the famous "imitation game" or, as it became known, the Turing Test. What interests me most about that paper is not the test, which assesses the extent to which a human would believe a machine providing calculated answers is actually another human,[51] but Turing's response to the various objections to or critiques of machine thinking. He dismissed objections based on theology, fear of the consequences, the seeming irreducibility of the question of consciousness, and others.[52] So will I.

Rather, I will focus on the constraints Turing himself took seriously. The first of these has to do with the relationship between intuition and logic. It turns out the *Entscheidungsproblem* exists even if you assume a mythical program, transcending logic and incorporating an otherwise unprogrammable insight, can be inserted into another program to calculate when it decides. The second is the acknowledgment that the *Entscheidungsproblem* would require an artificial intelligence, at *some* point for *some* problem, to rely on some mode of operation other than calculation, whether heuristic or random choice.

(a) Intuition

Once again, we assume that Kearse will be able to deal with *any* conceivable legal problem to the limits of what any machine can accomplish. Can all problems within machine-thinking be solved by way of the internal parameters of the thinking system, or is some form of exogenous—i.e., non-programmed—intuition going to remain necessary?

To answer this, we turn to Turing's next major work after *On Computable Numbers*, his 1938 Princeton Ph.D. dissertation, entitled *Systems of Logic Based on Ordinals*.[53] Hilbert's theses about the simultaneous consistency and completeness

[48] Turing, *Lecture*, *supra* note 1, at 394.
[49] *See infra* notes 68–79 and 133–135.
[50] Turing, *Intelligence*, *supra* note 12, at 441–64.
[51] *Id.* at 441–43.
[52] *Id.* at 449–58
[53] Jack Copeland, *Systems of Logic Based on Ordinals* (1939) [hereinafter Copeland, *Systems*], *in* THE ESSENTIAL TURING, *supra* note 1, at 146–204.

of mathematics had been demolished by Gödel, Turing, and others. Some problems were going to be undecidable within the bounds of formal logical systems. Turing's dissertation was an exploration of the kinds of problems "too hard" to be solved by a machine, even one with unlimited time and resources.[54] If a solution were available within the logical system, finding it might well be a matter of ingenuity. "Ingenuity" here has a limited meaning. It entails being creative with the rules of the system itself. But if the answer were not available within the system even with ingenious manipulation, the mathematician would have to resort to something else. We might call that non-systematic intuition or guesswork.

In *Systems*, Turing assumed, based on his own work and that of Gödel, some propositions were going to seem intuitively true to the mathematician, yet be undecidable as a matter of formal logic. The question he was addressing now was the *extent* to which ingenious manipulation of the logic of the formal system under consideration could replace non-systematic intuition. Without getting into the technicalities of the mathematical notation, we can simply accept that Turing posited a certain characteristic of logical functions within an ordinal system (a formal logical system consisting of a sequence of numbers[55]) as making them "number-theoretic." [56] As part of the project, he showed that the *Entscheidungsproblem* existed not only for a universal Turing machine, but also for problems that were not "number-theoretic," that is, not within the class of functions effectively calculable by a mechanical process in an ordinal system.

Turing posited the existence of a Turing machine capable of performing functions that were not number-theoretic, an example of which was the hypothetical halting program from *On Computable Numbers*. He called the program an "oracle" or *o*-machine, "a Turing machine augmented with one or more primitive operations each of which returns the values of some function (on the natural numbers) that is not Turing-machine-computable."[57] Recall that a Turing machine is simply a set of instructions as to state, input, operation given the input and state, and new state. The "machine" is reducible to a "satisfactory number" that can be coded onto a universal Turing machine. Being "satisfactory" means that the program will do its work to complete the calculation and then halt without going into a circle or loop. The hypothetical deciding program, H, as we have seen, leads to a contradiction in which it halts if it doesn't

[54] Jack Copeland, *Alan Turing 1912–1954*, *in* THE ESSENTIAL TURING, *supra* note 1, at 1.

[55] A set of ordinals is a sequence of numbers. One ordinal is first. Each ordinal has a successor. The set of ordinals is "well-ordered" if one of the ordinals is specified as the first one and each successor for each member is also specified. Joan Bagaria, *Set Theory*, STAN. ENCYCLOPEDIA PHIL. (Summer 2017 ed.), https://plato.stanford.edu /archives/sum2017/ entries/set-theory/.

[56] "Let l_1, l_2, l_3 . . . be any progression of logical systems indexed by (expressions for ordinals). To say that l_1, l_2, l_3 . . . is complete with respect to some set of formulae S is to say that for each formula x in S, there is *some* ordinal α such that x is provable in l_α." Copeland, *Systems*, *supra* note 53, at 140.

[57] B. Jack Copeland, *Turing's O-Machines, Searle, Penrose, and the Brain*, 58 ANALYSIS 128, 129 (1998).

halt, and vice versa. H is not a number-theoretic function because it is a logical impossibility.

The assumption of the *o*-machine, however, is that H works. That is, the section of the universal Turing machine running the *o*-machine (or program) will generate a 0 or 1 in the next square of the tape as though it did not run into the contradiction. H thus runs as an "oracle," meaning we don't know how, but manages to generate a 1 if the machine will halt and 0 if it will not.[58] Turing's assertion was that "it is not possible to construct an *o*-machine which, when supplied with the description of any other *o*-machine, will determine whether that machine is *o*-circle free. The argument may be taken over directly from [*On Computable Numbers*]." [59] By way of the same proof by contradiction, i.e., assuming that the *o*-machine leads to a true statement, Turing thus "completes the proof that problems of circle freedom of *o*-machines are not number-theoretic."[60]

What does this have to do with intuition and logic? What Turing initiated was an inquiry, still not wholly resolved, into the paradoxical idea that there are levels of undecidability. That is, in any given formally logical system, there may be statements that the mathematician senses to be true by intuition, but cannot prove by way even of ingenious use of the rules of inference within the system. Turing addressed the issue near the end of *Systems*. Before Gödel's undecidability proof, "it was thought by some that it would probably be possible to carry [Hilbert's] programme to such a point that all the intuitive judgments of mathematics could be replaced by a finite number of these rules. The necessity for intuition would then be entirely eliminated."[61] Turing had once again assumed that the rules existed in different systems by which intuition could be eliminated entirely, and mathematicians of unlimited ingenuity could discover them. He couldn't find one. The consequence was his belief that it would be impossible to find a formal logic that wholly eliminated intuition, and thus the need to "turn to 'non-constructive' systems of logic with which not all the steps . . . are mechanical, some being intuitive."[62] What his work on the logic of ordinal systems showed was that some methods of proof relied more or less on intuitive steps, that methods of proof should clearly denote when the steps were intuitive, and that it was unacceptable to be satisfied with these "non-constructive" proofs if it were possible to use methods that incorporated less undecidability.[63]

The *o*-machine showed that even making a first-order intuitive assumption (such as "the Turing machine m is circle-free") did not eliminate the need for higher order assumptions.[64] The mathematician Emil Post later expressed this as the concept of degrees of unsolvability. It seems intuitive to think of the original Turing machine *Entscheidungsproblem* as "easier" than the unsolvability of the

[58] *Id.* at 129–30.
[59] Turing, *Systems*, *supra* note 53, at 156–57.
[60] *Id.* at 157.
[61] *Id.* at 192–93.
[62] *Id.* at 193.
[63] *Id.*
[64] *Id.* at 142–43.

first order *o*-machine, in which we have assumed that there is a formal solution to the original *Entscheidungsproblem*. There is thereafter an infinite regress of unsolvability. That is, we could imagine a second order *o*-machine in which we have treated the problem of the first order *o*-machine as resolvable by a second order oracle. And so on. According to Post, in *Systems* "Turing proved a result which immediately generalizes to the result that for any "recursively given" unsolvable problem there is another of higher degree of unsolvability."[65]

Let's assume that Kearse is as sophisticated a thinking machine as can be built in the foreseeable future. It will still be a universal Turing machine. If it is a universal Turing machine, we cannot say that for *every* problem it will face, it will be able on its own to calculate when or even that it will be done thinking. In those circumstances, for Kearse to stop thinking and act, it will need to draw on something beyond its programming. To put the point as plainly as possible, if Kearse were to encounter one of those undecidable problems, it would not be able to decide it formally (i.e., through a sequence of validly number-theoretic steps) even stocked with a load of human intuition—the sense that something is true even if you cannot prove it formally.

I don't want to underestimate the ability of artificial intelligence developers to find pragmatic solutions to the theoretical constraints. In 1947, Turing said:

> It has for instance been shown that with certain logical systems there can be no machine which will distinguish provable formulae of the system from unprovable, i.e., that there is no test that the machine can apply which will divide propositions with certainty into two classes. Thus if a machine is made for this purpose it must in some cases fail to give an answer.[66]

That indeed is how programs for the verification of programs now appear to work. Being able to demonstrate formally that a piece of software works is valuable. Some programs can be shown formally to halt, but some cannot. No verification program would be able to be shown to work on every conceivable program, because that would resolve the *Entscheidungsproblem*. What a pragmatic verification program can be designed to say, at some point, is that the answer to the question whether the program being tested will halt is "unknown." In 2011, a review of developments in the field observed that the "challenge is . . . keeping the occurrences of the answer 'unknown' to within a tolerable threshold, in the same way that we hope Web browsers will *usually* succeed to download Web pages, although we know they will sometimes fail."[67]

According to Turing, what distinguishes a human mathematician reaching the conclusion of "unknown" (or "the question whether this program will halt is undecidable") is this: "[I]f a mathematician is confronted with such a problem

[65] *Id.* at 144 (quoting Emil L. Post, *Recursively Enumerable Sets of Positive Integers and their Decision Problems*, 50 BULLETIN OF THE AM. MATHEMATICAL SOC'Y 284, 289–90 (1944)).

[66] Turing, *Lecture*, *supra* note 1, at 393.

[67] Byron Cook et al., *Proving Program Termination*, 54 COMM. OF THE ACM 88, 90 (2001).

[(the machine failing to give an answer)] he would search around a[nd] find new methods of proof, so that he ought eventually to be able to reach a decision about any given formula."[68] Whether that is by way of intuition, creativity, inspiration, or divine guidance, it seems to me sensible to conclude it will be the hardest thing to program into Kearse.[69] I am not suggesting it could not be done; Turing suggested deliberately having the machine give a wrong answer than no answer, so as to learn from its mistake.[70] But not having a *Entscheidungsproblem* is what we human lawyers appear to have over Kearse for a while at least.[71]

(b) Mistakes and Heuristics

The objection to the imitation game Turing needed most to meet on its own terms, since his resolution of the *Entscheidungsproblem* was the source of it, was the "mathematical objection." He put aside obviously normative questions that might be posed to the machine, such as, "What do you think of Picasso?" Rather,

> [t]he questions that we know the machines must fail on are of this type, "Consider the machine specified as follows . . . Will this machine ever answer 'Yes' to any question?" The dots are to be replaced by a description of some machine in a standard form, which could be something like [any digital computer that is a universal Turing machine].[72]

Note the context of the observation. Turing was asking how a machine could win the imitation game if asked a question that it was theoretically incapable of answering. The issue was whether the machine's inability to answer the question

[68] Turing, *Lecture*, *supra* note 1, at 393–94.

[69] For an extended treatment of the relationship of logic to creativity in mathematics that follows on Turing's distinction between ingenuity and intuition, see WILLIAM BYERS, HOW MATHEMATICIANS THINK: USING AMBIGUITY, CONTRADICTION, AND PARADOX TO CREATE MATHEMATICS (2007).

> [T]here are two visions of mathematics that seem to be diametrically opposed to one another. These could be characterized by emphasizing the "light of reason," the primacy of the logical structure, on the one hand, and . . . a creative light that I maintain often emerges out of ambiguity, on the other (this is itself an ambiguity!). My job is to demonstrate how mathematics transcends these two opposing views: to develop a picture of mathematics that includes the logical and the ambiguous, that situates itself equally in the development of vast deductive systems of the most intricate order and in the birth of the extraordinary leaps of creativity that have changed the world and our understanding of the world.

Id. at 11–12. For an entertaining piece of fiction that invokes the gift and (spoiler alert) loss of the gift of creative and transformative mathematics, see REBECCA GOLDSTEIN, THE MIND-BODY PROBLEM (1993).

[70] Turing, *Lecture*, *supra* note 1, at 394.

[71] But if you believe the computational theory of mind is correct, you should disagree with this assertion. *See infra* notes 85 and 87. As to Turing's own view, see Piccinini, *supra* note 47, at 39 ("[Turing] was fond of saying that no Turing machine could correctly answer all mathematical questions, while human mathematicians might hope to do so.").

[72] Turing, *Intelligence*, *supra* note 12, at 451.

would cause it to lose the imitation game, i.e., to make responses that were distinguishable from a human's.

Turing's answer was not to deny the machine's theoretical constraint, but to ask whether similar conditions also applied to human intellect.[73] When you are in the midst of thinking through a problem and, for whatever reason, have not reached what you would consider a conclusive result, you may resort to a "rule of thumb" or "heuristic." A heuristic may not give you the perfect answer, but it shortens the time for getting one. The complement to using heuristics, and possibly making a mistake, is learning from the mistake. Hence, a machine that could win the imitation game might well be one that uses heuristics, makes mistakes, and learns from the mistake in the same way that a human expert might.

Here is an example of one of my own heuristics from management of an in-house law practice. We were resource-constrained, with one general lawyer per business unit (assume each business unit is roughly $1 billion in annual sales and 1,000 employees). Many people, particularly in the sales and purchasing functions, wanted the lawyer to review contracts. There were only so many hours in a day. I said words to the effect, "If the contract involves less than $100,000 or is terminable in fewer than ninety days, don't spend time on it." That is a heuristic, not an algorithm. Would it have been possible to make a mistake and not review an important million-dollar contract that had only a thirty-day life? Sure. But not doing so ought not to have been a fatal error, only a way of developing a better heuristic for deciding which contracts to review.

Turing's essays on this point from the late 1940s and early 1950s are prescient on the issue of theoretical possibilities of machine learning.[74] He saw no reason why machines could not come to learn by making mistakes and correcting them. Indeed, some of the breakthroughs in the Enigma code-breaking during World War II came from the use of heuristics that significantly reduced the amount of processing the code-breaking "bombe" machines need to do.[75] Making mistakes is at the heart of the relationship between machine learning and human learning. Even if the machine has intuition, the *o*-machine proof shows it is still limited by the *Entscheidungsproblem*. A human is not so limited: "[O]ne can show that however the machine is constructed there are bound to be cases where the

[73] *Id.*

[74] A primary issue for Turing and others almost seventy years ago was not theoretical capability but physical storage capacity. In 1947, the Manchester computer on which Turing was working had storage capacity of about 200,000 (2×10^5) binary digits, or, as Turing described it, "the memory capacity of a minnow." Turing, *Lecture*, *supra* note 1, at 383. Turing cited estimates of the storage capacity of the human brain as being between 10^{15} and 10^{20} binary digits (100,000,000,000,000,000,000), and predicted that something on the order of 10^9 binary digits would be enough for satisfactory playing of the game. Turing, *Intelligence*, *supra* note 12, at 459. Just to get a sense of where we are today, the MacBook Pro on which I am typing this has a 500-gigabyte hard drive. That is 4,294,967,296,000 or about 4×10^{12} binary digits of storage.

[75] Jack Copeland, *Artificial Intelligence*, *in* THE ESSENTIAL TURING, *supra* note 1 ("Copeland, *Artificial Intelligence*"), at 353–55.

machine fails to give an answer, but a mathematician would be able to."[76] But, assuming no mechanical failure, the machine cannot make a mistake. The key is that "this danger of the mathematician making mistakes is an unavoidable corollary of his power of sometimes hitting upon an entirely new method."[77] Turing's own early assessment of a real learning machine (i.e., one that taught itself from its own mistakes rather than being reprogrammed into a new machine by a human) was that it would need to use "crude rules of thumb" and ought to be programmed to make occasional random choices from which it could make mistakes and learn.[78] If so, Turing contended, "machines can be constructed which will simulate the behaviour of the human mind very closely."[79]

II. Thinking and Acting for Machines and Humans

A. Lawyer-Humans and Lawyer-Automatons

Let's review where we are now. I am willing to assume the proliferation of the most sophisticated Kearses that are not, at this moment, mere fantasies. Put otherwise, I will accept the Susskinds' thesis of increasingly capable machines. Indeed, I posit them as the most capable. Having said that, I do not believe even they anticipate computational abilities beyond the most capable *Turing machines*, even if the deep learning capabilities they do anticipate run on the most capable Turing machines.[80]

I want to be clear on the raft of controversial issues that I am not addressing here. They include: (a) whether Kearse, a lawyer-automation with AI capability, could ever appear to replicate human-level intelligence; (b) whether the Halting Problem means that such replication is impossible; (c) whether the machine is really "thinking"; (d) whether the Halting Problem has some bearing on the scientific basis for consciousness; or (e) whether an artificially intelligent entity has the right to legal personhood.[81] For now, I am limiting myself to the proposition that Kearse is not capable of deciding itself, for *every* program and *every* situation it might face, that the program it was running was in an infinite loop and would not return an answer.[82]

[76] Alan Turing, *Intelligent Machinery, A Heretical Theory*, [hereinafter Turing, *Heretical Theory*], *in* THE ESSENTIAL TURING, *supra* note 1, at 472.

[77] *Id.*

[78] *Id.* at 474–75. For a similar conclusion about Turing's response to the mathematical objection, *see* Piccinini, *supra* note 47, at 37–38.

[79] *Id.* at 472.

[80] SUSSKIND & SUSSKIND, *supra* note 2, at 186–87.

[81] *See, e.g.*, Lawrence B. Solum, *Legal Personhood for Artificial Intelligences*, 70 N. CAR. L. REV. 1231 (1992).

[82] I have taken to heart Melanie Mitchell's caution to me not to repeat the common misstatement of the Halting Problem. Email from Melanie Mitchell to Jeffrey Lipshaw, Aug. 2, 2017 (on file with the author). There *are* indeed programs intended to do the job of determining whether other programs will halt—the field of formal verification. The confusion arises as between "falsely believ[ing] we are always unable to prove termination, rather than more benign consequence that we are unable to always prove termination." Cook, et al., *supra* note 67, at 88.

My thought experiment here is to put aside all of those issues and focus the issue, as between lawyer-humans and lawyer-automatons, of deciding to decide. The one fundamental but unresolved philosophical issue that *does* bear on the question is whether there is something about the conscious human mind that goes beyond mere computation.[83] The literature on the subject is abundant. There is an acknowledged split on this issue as between the AI community and other disciplines in the humanities and social sciences.[84] If you limit yourself to the AI community, it is fair to say that the predominant view will be that the computational model of the mind ought to prevail. If that's the case, then humans are also subject to the Halting Problem, and the whole subject is either irrelevant or a red herring. Two eminent AI theorists have expressed that sentiment to me.[85] But I am comfortable that there is literature in the AI community to the effect that something like the Halting Problem bears on the theoretic constraints to my lawyer-automaton Kearse. Moreover, once you extend beyond the AI community, and get other disciplines (particularly philosophers) involved, there are even richer critiques of the computational theory of mind.[86]

I am most interested here in the implications, if any, of the mathematical limitations on minds *or* machines that can only compute as well as the most capable Turing machines.[87] For example, the biologists George Reeke and Gerald

[83] *See* Michael Rescorla, *The Computational Theory of Mind*, STAN. ENCYCLOPEDIA PHIL. (Spring 2017 Edition), https://plato.stanford.edu/archives /spr2017/entries/ computational-mind/.

[84] *See Introduction*, in MECHANICAL BODIES, COMPUTATIONAL MINDS: ARTIFICIAL INTELLIGENCE FROM AUTOMATA TO CYBORGS 1 (Stefano Franchi & Güven Güzeldere eds., 2005) [hereinafter FRANCHI & GÜZELDERE, MECHANICAL BODIES].

[85] Professor Mitchell told me she is thoroughly convinced the Halting Problem has no relevance to artificial intelligence. She does not disagree with my statement of the Halting Problem, i.e., that Kearse won't be able to determine whether a program halts for any such possible program it might run. But she points out, neither can humans. Humans can give up on the problem, and can program machines to do so as well, to flip a coin or resort to eeny-meeny-miny-mo. Email from Melanie Mitchell to Jeffrey Lipshaw, Aug. 2, 2017 (on file with the author). Nevertheless, it is clear that Professor Mitchell believes there are other reasons for being circumspect about the prospect of human-level AI. *See infra* notes 115–117 and accompanying text. Professor Laird was equally dismissive, telling me there is no plausible theory of the human mind other than it is ultimately computational, and hence the Halting Problem issue is a red herring. Email from John Laird to Jeffrey Lipshaw, Aug. 10, 2017 (on file with the author).

[86] *See generally* the essays collected in FRANCHI & GÜZELDERE, MECHANICAL BODIES, *SUPRA* NOTE 84.

[87] The physicist Roger Penrose has written two controversial books making the argument to the effect that mathematical limitations of computation have something to do with the difference between AI and the conscious human mind. Penrose, *supra* note 21; ROGER PENROSE, SHADOWS OF THE MIND (1994). Others have refuted his "Gödelian" or "Halting Problem" arguments. *See* Selmer Bringsjord & Hong Xiao, *A Refutation of Penrose's Gödelian Case Against Artificial Intelligence*, 12 J. EXPERIMENTAL & THEORETICAL ARTIFICIAL INTELLIGENCE 307 (2000). Note, however, that Professor Bringsjord's refutation arises from a criticism of Penrose's math, not a belief either (a) that Penrose is wrong about the human mind as constituting something beyond mere computation, or (b) that something like Gödel's incompleteness theorem or the

Edelman noted in 1988 that even the most impressive AI demonstrations "share a common quality that John McCarthy [the Turing Award winner who developed the AI LISP programming language[88]] and others have repeatedly pointed out: they are 'brittle' in the sense that, if pressed around the edges, they tend to 'crack.' In other words, the programs lack commonsense knowledge and reasoning—they do not 'know' their own limitations."[89] One consequence of the brittleness is overcoming the "unending compounding" of "exceptions to exceptions that are present in real life situations" and thus require something like common sense.[90] Hence, Reeke and Edelman concluded:

> This unending compounding of exceptions comes close to revealing the true nature of the brittleness problem, which is that no amount of anticipation, planning, and programming can ever enumerate, a priori, all the variants of even a routine situation that may occur in daily life. It would seem, in what is perhaps an analogy to the classic "halting problem" in the analysis of algorithms (the problem of determining whether a given algorithm will run forever or eventually halt), that the only way to determine all the responses a system needs to have to deal with the vagaries of the real world is to expose it to the world and let it "run." Thus, each system will be different and fundamentally unprogrammable. [91]

In the same vein as this symposium's titular dark allusion and invocation of Ray Kurzweil's "singularity," in 2000, Bill Joy, the co-founder of Sun

Halting Problem has a bearing on the issue. The Bringsjord and Hong article concludes: "The upshot is that the future, with respect to Penrose's goal, is still open. One of us (Bringsjord) is in complete and utter agreement with Penrose that it *is* possible to derive the denial of "Strong" AI from Gödelian facts, and is working on producing the demonstration." *Id.* at 326. But if you believe that, ultimately, the computational theory of mind will prevail, then if a machine has a Halting Problem, so does a human. In that case, the issue would indeed be irrelevant to AI.

[88] Daniel C. Dennett, *When Philosophers Encounter Artificial Intelligence*, 117 DAEDALUS 283, 283 (1988).

[89] George N. Reeke, Jr. & Gerald Edelman, *Real Brains and Artificial Intelligence*, 117 DAEDALUS 143, 149 (1988).

[90] *Id.* at 152. Obviously, the more developed the program, the less cracking and brittleness there will be. That is the goal of the most advanced AI theory seeking to replicate intelligent behavior through "computationally-realizable mechanisms and structures that can answer all the questions we might want to ask about cognitive behavior." Jill Fain Lehman, John Laird, & Paul Rosenbloom, *A Gentle Introduction to Soar, an Architecture for Human Cognition: 2006 Update 2* (2006), http://web.eecs.umich.edu/~soar/sitemaker/ docs/misc/GentleIntroduction-2006.pdf. The Soar AI architecture, for example, seeks to overcome AI brittleness and inflexibility by changing the algorithms used to update the system's "beliefs" about the environment in which it is working. Soar Technology, *Soar: A Comparison with Rule-based Systems* (2002), http://web.eecs.umich.edu/~soar/sitemaker/docs/misc/SoarRBSComparison.pdf. The most up-to-date description of the Soar technology is JOHN E. LAIRD, THE SOAR COGNITIVE ARCHITECTURE (2012).

[91] Reeke, *supra* note 90.

Microsystems, published in *Wired* a now-famous and pessimistic prediction of the "coming of the machines."[92] In 2008, Selmer Bringsjord, the chair of the Department of Cognitive Science at Rensselaer Polytechnic Institute, a specialist in "the logico-mathematical and philosophical foundations of artificial intelligence,"[93] responded.[94] One basis for Joy's doomsday prediction was that advances in robotics, along with the processing speeds predicted by Moore's Law, would permit humans to download themselves out of their bodies and into more durable brains or bodies. In addition to questioning whether Joy's computational model of the mind was a fair assumption, Professor Bringsjord rebutted the idea that the hypothetical robot—a universal Turing machine—could run a human-like (or oracle) program merely because it had more computing power. In other words, no matter how much storage capacity and speed, the robot could not compute a Turing-uncomputable function like the Halting Problem.[95]

What suggests to me that human brains aren't Turing machines is the evidence that a lawyer-human like me has the freedom, unconstrained by computational limitations, *always* to be able to decide to decide.[96] If I turn my iPhone calculator, a Turing machine, sideways and ask it to calculate the cube root of 502, it returns this number almost instantaneously: 7.947573854561911. The calculator doesn't give me an exact answer, because the cube root of three is irrational. It is not a ratio of natural numbers; it will extend infinitely; and it will not display any pattern. The calculator uses an algorithm with a finite number of

[92] Bill Joy, *Why the Future Doesn't Need Us*, WIRED (Apr. 1, 2000), https://www.wired.com/2000/04/joy-2/. Indeed, in the first paragraph of his essay, Joy attributes his pessimism to a conversation with Ray Kurzweil.

[93] Selmer Bringsjord, RENSSELAER POLYTECHNIC INST. (2016), http://www.hass.rpi.edu/pl/faculty-staff-s17/selmer-bringsjord (faculty page).

[94] Selmer Bringsjord, *Ethical Robots: The Future Can Heed Us*, 22 AI & SOC'Y 539 (2008).

[95] The passage bears quoting in full:

> Let *f* be a function from the natural numbers to natural numbers. Now suppose that the storage capacity and speed of today's computers grow for 1,000 years at rates that exceed even what Joy has in mind; and suppose, specifically, that C is the best computer available in 3006. Question: Does it follow that C can compute *f*? No, of course not, and the proof is trivial: simply define *f(n)* to be the maximum productivity of n-state Turing machines with alphabet {0, 1}, where these machines are invariably started on an empty tape, and their productivity corresponds to the number of contiguous 1s they leave on the tape, after halting with their read/write head on the leftmost of these 1s. Since this famous function, so-called Σ or "busy beaver" function, is Turing-uncomputable[,] . . . C, no matter how fast, cannot compute *f*. *(Of. [sic] course, any Turing-uncomputable function will do. E.g., the halting problem would do just fine.)* . . . Speed is great, but however fast standard computation may be, it is still by definition at or below the Turing Limit.

Id. at 546–47 (emphasis added).

[96] And there is a rich AI literature since Turing on the possibilities of free will in automatons. *See, e.g.*, John McCarthy, *Free will—even for robots*, 12 J. EXPERIMENTAL & THEORETICAL ARTIFICIAL INTELLIGENCE 341 (2000); Lei Liu, *On Artificial Intelligence & Free Will*, 6 J. CONSCIOUSNESS EXPLORATION & RESEARCH 496 (2015).

rules to approximate the answer to as many decimal places as the calculator provides (several more if you turn the iPhone sideways than if you hold it straight up). What the calculator could never do, even with infinite capacity á la Bill Joy's doomsday robot, is determine whether it would ever reach a conclusion whether the cube root of 502 is computable. In the formal system, that question is undecidable. Absent some external limitation, as Reeke and Edelman suggest, the calculator would just keep calculating more and more digits, with no ability to stop, and no ability to know whether it would ever stop. Again, not to overstate the point, digital computers can be programmed to verify the terminability of many programs, or to say at some point effectively, "I quit because I don't know," or, "This isn't working; do something else," and to do the latter "infrequently enough that they are useful in practice."[97] But no digital computer on its own could always, for *every* program, decide to decide.

I'm not a digital computer. As Turing understood, I have intuitions that are not provable in a formal logical system in the same way that a mathematician has such intuitions. A mathematician could do the same cube root calculation by hand. The mathematician would, at some point (assuming she were sane), decide to stop. But she would know intuitively that she could continue to generate infinitely more digits after the decimal place without going into a loop; i.e., that the number she was calculating was computable. I have the same ability as a lawyer. I can't prove whether my dithering ultimately loops or not. I suspect it may. Nor can I prove whether my sense of free will is an illusion. But I am not going to wait to find out.

B. The Neural Network Red Herring

Could *some* machine get beyond the Halting Problem and *always* be able to decide to decide? The cutting edge of the mathematics of computing today is called "hypercomputation" or computation beyond the "Turing Limit." It involves the theoretical and practical issues of computing, but not bounded by the limitations of present technology, namely, the digital computer.[98] For example, mathematicians have proved that a theoretical universal *analog* computer (i.e., one that is not required to operate in discrete steps on discrete numbers like a Turing machine) can solve the *Entscheidungsproblem* (i.e., determine whether it will always be able to decide a problem).[99] The difference is that Turing machines are based on sequences of discrete numbers. Metaphorically, think of the numbers in a Turing machine as pennies in a roll or links in a chain. They have an order to them, and may extend infinitely, but you can jump from one to the next. But when you jump, say, from 1 to 2, there are infinite numbers in between them that may or may not be computable. And no matter how many places to which you extend

[97] Cook, et al., *supra* note 67.

[98] *See, e.g.*, Francisco Antonio Doria, *Blueprint for a Hypercomputer*, *in* A Computable Universe: Understanding and Exploring Nature as Computation 333–44 (Hector Zenil ed., 2012); B. Jack Copeland & Richard Sylvan, *Beyond the Universal Turing Machine*, 77 Australasian J. Phil. 46 (1999).

[99] *See* Bruce J. MacLennan, *Transcending Turing Computability*, 13 Minds & Machines 3 (2003).

two numbers after the decimal point, there is still the possibility of more numbers in between.[100] All of those infinitely possible numbers, computable or not, taken together are "real numbers." The proper metaphoric conception for real numbers would not be a linked chain, but a continuous string that could be sliced anywhere, and capable of infinitely more precise slices between any two previous slices.[101]

Which leads us to the current state of artificial intelligence and theoretical possibilities of "hypercomputation" that go beyond what a universal Turing machine can do. When you talk about the theoretical constraints on increasingly capable machines, it's likely somebody is going to say something like, "Well, what about 'neural networks,' 'latent semantic analysis,' 'neural probabilistic language models,' and 'deep learning'? Computers can now do things like facial recognition and have other human-like learning capabilities." We are going to assume Kearse has the most theoretically advanced of any such capabilities, but we are not going to engage in science fiction.

The short answer is that the Halting Problem still applies. Even running on a presently conceivable neural network (the most complex of which is called a "deep learning" program), as opposed to the simple Turing machine scanner, Kearse will not be able to determine, while in the process of solving every problem, that every problem is solvable. The longer answer involves understanding what all these terms mean. The basic idea is that the machine is a collection of thousands or millions of interconnected nodes—each of which is a simple processor.[102] These networks of nodes were called "neural" because early researchers viewed them as idealized models of biological neurons.[103] Each node accepts information either from an outside source or from another node. There is no magic here, however. A neural network is different from a serial processor (idealized by a Turing machine) because there is no single processor that "is" either the universal machine or a program within it. In the simplest possible terms, the network accepts information in input nodes, weights it, compares, processes it, and sends it to "hidden" layers of other nodes, over and over again, and finally produces results in output nodes.[104] Is this a picture of a little girl or a coyote? The artificial neural network will take a stab at it, remember its mistakes (when it identified a girl as a coyote or vice versa), and learn from them. "Deep learning"

[100] Take the following two numbers which are very close: 4.23950606865443 and 4.23950606865444. The only difference is the last digit. The following two numbers are between those two: 4.239506068654435 and 4.239506068654436. I have added one more decimal place to the smaller number, and inserted into it 5 and 6, respectively. I could keep repeating this exercise infinitely.

[101] *See* Turing, *On Computable Numbers*, *supra* note 21.

[102] *See* Jürgen Schmidhuber, *Deep Learning in Neural Networks: An Overview*, 61 NEURAL NETWORKS 85, 86 (2015).

[103] Rescorla, *supra* note 83. One of the thinkers who first considered the possibilities of neural networks was none other than Alan Turing himself. A.M. Turing, *Intelligent Machinery* [hereinafter Turing, *Intelligent Machinery*], *in* THE ESSENTIAL TURING, *supra* note 1, at 410–32. *See also* Copeland, *System*, *supra* note 54, *in* THE ESSENTIAL TURING, *supra* note 1, at 402–03.

[104] *See* Schmidhuber, *supra* note 102, at 86–87.

is shorthand for a neural network with many, many, many layers, and hence the ability to process and learn even more "deeply."[105]

Despite the interconnected operation of any presently buildable *artificial* neural network, it still must operate within the extant modes of computation on discrete digits using discrete steps. As has been observed, "every [artificial] neural network ever physically constructed has been implemented on a digital computer."[106] The point of Turing's *o*-machine exercise was to show that even a Turing machine that incorporated some aspect of hypercomputational ability would still be subject to the theoretical limitations, i.e., the Halting Problem. The machines we are now talking about would have to be something wholly other than a universal Turing machine.[107]

I want to put aside philosophical and scientific debates about intuition that post-date Turing's work. For Turing, intuition was simply that capability of a human mathematician to sense that a mathematical statement was true even though it could not be proved formally (i.e., by means of an algorithm in discrete steps).[108] The question was not whether mathematical problems (including puzzles) were solvable by any means including intuition, but whether one could prove, formally and non-intuitively, there was an algorithm that would lead to the solution of the puzzle.[109] Presently, discussions about human intuition go far beyond the limits of formal systems. The skeptical critique *par excellence* of human intuition is Daniel Kahneman's popular *Thinking Fast and Slow*.[110] In Kahneman's assessment, intuition is largely the product of what he calls "fast" or "System 1" thinking: associative rather than logical short cuts or heuristics that get us to conclusions quickly, but often with significant errors.[111] If experts have

[105] *Id.* at 86:

> *Learning* or *credit assignment* is about finding weights that make the [neural network] exhibit *desired* behavior, such as driving a car. Depending on the problem and how the neurons are connected, such behavior may require long causal chains of computational stages[,] . . . where each stage transforms (often in a non-linear way) the aggregate activation of the network. Deep Learning is about accurately assigning credit across *many* such stages.

[106] Rescorla, *supra* note 83 (I have added the bracketed word for clarity).

[107] *See supra* notes 54–65 and accompanying text.

[108] As noted above, his Ph.D. dissertation explored the extent to which formal proof could replace intuition. *See* Copeland, *supra* note 61. A good example is Fermat's Last Theorem. *See* SINGH, *supra* note 15. For centuries, it *seemed* like it was true, and nobody could find an example of three positive integers, x, y, and z, for which $x^n + y^n = z^n$ for any integer n greater than 2. But nobody *proved* it until Andrew Wiles came along. The difference between mathematical intuition and proof is the central plot issue in the recent film about the short career of the brilliant Indian mathematician Srinivasa Ramanujan, THE MAN WHO KNEW INFINITY (Edward R. Pressman Film, 2015).

[109] *See* Alan Turing, *Solvable and Unsolvable Problems*, 31 SCI. NEWS 7 (1954), *in* THE ESSENTIAL TURING, *supra* note 1, at 592–93.

[110] DANIEL KAHNEMAN, THINKING FAST AND SLOW (2011).

[111] *Id.* at 20–24. I have previously written about the infinite regress of assessing one's own biases and heuristics. *See* JEFFREY M. LIPSHAW, BEYOND LEGAL REASONING: A CRITIQUE OF PURE REASONING 128–29 (2017); Jeffrey M.

trustworthy "intuitions," it is not the result of any magic, but of long experience and deep learning by which the expert recognizes patterns quickly.[112] Others acknowledge the elusiveness of trying to make a rigorous science out of the sources of expert intuition, but recognize (a) how it differs from machine-like algorithmic thinking, and (b) its centrality to problem-setting, problem-solving, and decision-making. If constructive intuition brings anything to the party, it is by way of metaphor or analogy in which the problem-solver sees in the new pattern something similar in an old pattern.[113]

Could Kearse be a thinking machine that has intuitive capabilities beyond a Turing machine? Perhaps. Whether that will ever occur, however, is an empirical question, presently answerable, if at all, solely by inductive reasoning, not formal proof. To answer "yes," i.e., that *any* Kearse could teach itself to find *all* the answers by ingenious (or patient) manipulation of its own parameters, appears to be an inductive conclusion Turing himself warned against making.[114]

Thus, it is possible that the Halting Problem is irrelevant if an analog rather than digital Kearse were available. But assuming the existence of an analog Kearse assumes away the problem, because presently the only extant rather than theoretical *universal* analog computer of which we are aware is a biological brain. The key distinction, even in the mathematics, between the discreteness of digits and the continuity of analogs, does not surprise me. A computer has a much harder time than a human making a meaningful analogy.[115] Melanie Mitchell's Ph.D., supervised by Douglas Hofstadter, involved attempt to write a digital computer program "that could make human-like analogies in its microworld."[116] Hence, I take seriously Professor Mitchell's own intuitions about the relationship of the digital and analog worlds when it comes to Kearse's potential capability:

> The ultimate goal of AI is to take humans out of the *meaning* loop and have the computer itself perceive meaning. This is AI's hardest problem. The mathematician Gian-Carlo Rota called this problem "the barrier of meaning" and asked whether or when AI would ever "crash" it. I personally don't think it will be anytime soon, but if and when this barrier is unlocked, I suspect that analogy will be the key.[117]

Lipshaw, *Dissecting the Two-Handed Lawyer: Thinking Versus Action in Business Lawyering*, 10 BERKELEY BUS. L. J. 231, 242–43 (2014).

[112] *Id.* at 239.

[113] *See, e.g.*, WILLIAM BYERS, THE BLIND SPOT: SCIENCE AND THE CRISIS OF UNCERTAINTY 1–16 (2011); BYERS, *supra* note 69; GARY KLEIN, SOURCES OF POWER: HOW PEOPLE MAKE DECISIONS 285–93 (1999); GARY KLEIN, THE POWER OF INTUITION 3–12 (2003); DONALD A. SCHÖN, THE REFLECTIVE PRACTITIONER: HOW PROFESSIONALS THINK IN ACTION 182–87 (1983).

[114] *See* Copeland, *Systems*, *supra* note 53, at 193.

[115] MITCHELL, *supra* note 14, at 187.

[116] *Id.* at 193.

[117] *Id.* at 208. For my own assessment of the role of metaphor and analogy for the creation of meaning in the process of lawyering, see LIPSHAW, *supra* note 111, at 149–55.

If a hyper-computational super-Kearse is possible, it is still the one about which we presently have the least to worry.[118] So we should be focusing on teaching and using the most human and least *digital* machine-like capabilities. What are they?

C. Deciding and Acting Under Uncertainty

I want to be circumspect about what I am claiming lawyer-automatons are least able to do. Kahneman and several co-authors have recently made a nuanced argument for letting algorithms take the place of human professional predictions or decisions *up to a point*.[119] The issue is not bias, which might affect the accuracy of a judgment, but "noise" variability that affects the reliability of a judgment. A biased judgment may be consistently off-target (i.e., inaccurate); a "noisy" judgment is scattered (i.e., unreliable).[120] Their examples of instances of noisy judgments include underwriting insurance decisions, credit ratings, stock valuations, real estate appraisals, criminal sentencing, job performance evaluations, emergency room evaluations, and financial statement audits. To be fair, the authors do not suggest that the use of algorithms can substitute for all professional judgment. They merely contend that managers are generally unaware of the level of the noise, and it is far above what they would consider acceptable. In addition to better accuracy, "the key advantage of algorithms is that they are noise-free: Unlike humans, a formula will always return the same output for any given input. Superior consistency allows even simple and imperfect algorithms to achieve greater accuracy than human professionals."[121]

The question left open (indeed, begged) is the point to which algorithms either can or should replace human judgment. Kahneman and his co-authors, I suspect, are suggesting that calculated judgments *should* replace human judgments whenever they can do so as a practical matter. The areas of impracticality they concede still exist include those in which the data inputs are idiosyncratic or hard to code, or when the decision involves multiple dimensions or depends on a negotiated outcome.[122] And they have no answer for these infinite regresses:

> Of course, no matter what type of algorithm is employed, people must retain ultimate control. Algorithms must be monitored and adjusted for occasional changes in the population of cases. Managers must also keep an eye on individual decisions and have the authority to override the algorithm in clear-cut cases. For example, a decision to approve a loan should be provisionally reversed if the firm discovers that

[118] *See* NEUMANN, *supra* note 46.

[119] *See* Daniel Kahneman, Andrew M. Rosenfeld, Linnea Gandhi, & Tom Blaser, *Noise: How to Overcome the High, Hidden Cost of Inconsistent Decision Making*, HARV. BUS. REV. 39 (Oct. 2016).

[120] *See id.* at 41–42

[121] *Id.* at 41.

[122] *See id.* at 44.

> the applicant has been arrested. Most important, executives should determine how to translate the algorithm's output into action.[123]

Why assume no algorithm of deep learning, in which the machine adjusts the algorithm for changes in population? If so, then what is the algorithm for deciding what is a clear-cut case for overriding the algorithm? Most importantly, what is the algorithm for determining that the algorithm has done its work or cannot do its work, and it is time to take action?[124]

In any event, I want to grant Kearse every conceivable ability that a machine running on algorithms can have. I will assume that a digital Kearse capable of deep learning may well have something that appears to be intuition that is indistinguishable from that of a human lawyer. It may be able to teach itself heuristics that shorten the process of thinking. I am skeptical of arguments based purely on lack of trust in the machine.[125] The example to which I return over and over is the relationship of an airline pilot to the aircraft's Traffic Collision and Avoidance System (TCAS).[126] This is a computerized system, mandated by the FAA for all aircraft with more than a certain number of seats, and for any aircraft flying within busy airspace. The system coordinates transponders in each aircraft, warning the pilots of encroachments into the area surrounding it, and issues complementary instructions to the aircraft to avoid collisions. The instructions include "Traffic Advisories" and more serious "Resolution Advisories" for which the pilots are given complementary instructions to ascend or descend. The only instruction to pilots with higher priorities than Resolution Advisories are those that come from the systems monitoring wind shear and proximity to the

[123] *Id.* at 48.

[124] For a discussion of the infinite regress (i.e., the non-algorithmic or discontinuous aspect) of creativity in mathematics itself, see WILLIAM BYERS, DEEP THINKING: WHAT MATHEMATICS CAN TEACH US ABOUT THE MIND 80–81, 129–30 (2015). Professor Byers observes:

> What, in particular, are the consequences of my assertion that there is no possible algorithm, rule, or technique for creativity? One consequence, which would be unacceptable to the artificial intelligence community, is that no one will ever program a computer to be creative on its own, that is, independent of human intervention. You might succeed in simulating creativity through a new idea in artificial intelligence which would itself arise out of a creative act on the part of a computer scientist. This kind of breakthrough might be useful and have all kinds of theoretical and economic consequences. Nevertheless, there remains an unbridgeable gap between a simulation of creativity and the real thing.

Id. at 122–23. That seems right to me: not to be overly dramatic about it, but if *everything* is reducible computationally in a Turing machine, and a universal Turing machine can compute anything that is computable, then what is the source of the First Goal and the choice of the First Algorithm directed to it?

[125] As are the Susskinds. *See* SUSSKIND & SUSSKIND, *supra* note 2, at 39–40, 233–38.

[126] *See* FED. AVIATION ADMIN., INTRODUCTION TO TCAS II VERSION 7.1 (Feb. 28, 2011), https://www.faa.gov/documentLibrary/media/Advisory_Circular/TCAS%20II%20V7.1%20Intro%20booklet.pdf.

ground.[127] Due to TCAS and other traffic control measures, there was a "five-fold decrease in mid-air collisions per flight hour" between 1970 and 2014.[128] TCAS is a perfect example of a system that uses algorithms to replace human perception. I wouldn't be a passenger on a plane that lacks it, even if many pilots argue that pilot-less aircraft (unlike driver-less cars) are not going to happen any time soon.[129] I believe human trust in machines develops and evolves over time, such that we no longer blink at doing things our ancestors might have feared (like flying or going up 100 stories in an elevator).

It is important to remind ourselves again that Kearse, as the most advanced possible Turing machine, doesn't do anything that a human being with a pencil and paper can't do; it just does it faster and has access to information on a scale that far exceeds any human.[130] It may appear to be playing football or drawing pictures or interpreting appellate opinions, but it is calculating with numbers. Even when it appears to draw inductive, probabilistic, or intuitive conclusions, it is operating within a framework of formal, deductive logic. Thus, assuming Kearse is a being, it is the most rational of all possible beings. The claim here is merely this: because of the Halting Problem, Kearse can never be shown to have the ability, under every conceivable program running within it, to decide on its own that the program will halt. If it is a universal Turing machine, there will be a program for which it cannot decide that it will decide. For the foreseeable future, what a human will be able to do better than a digital computer is *always* being able to decide to act, regardless of the possibility of a logical solution, before the calculation is complete.

Nor do I want to get bogged down in endless and unresolvable debates about whether *any* algorithm-based machine of *any* kind could *ever* replicate human

[127] *See id.* at 35–36.

[128] David DeLaGarza, *TCAS – The Last Line of Defense Against Midair Collisions,* Airline Reporter (June 25, 2014), http://www.airlinereporter.com/2014/06/tcas-the-last-line-of-defense-against-midair-collisions/.

[129] *See* Alan B. Chambers & David C. Nagel, *Pilots of the Future: Human or Computer?*, 28 Communications of the ACM 1187 (1985); Haitham Baomar & Peter J. Bentley, *An Intelligent Autopilot System that Learns Flight Emergency Procedures by Imitating Human Pilots*, 2016 IEEE Symposium Series on Computational Intelligence (SSCI) 1, http://ieeexplore.ieee.org/abstract/document/7502578/; John Markoff, *Plane Without Pilots*, N.Y. Times, at D1 (Apr. 6, 2015); Patrick Smith, *We Are Told That Planes Basically Fly Themselves. How True is This?*, Ask the Pilot, http://www.askthepilot.com/questionsanswers/automation-myths/ (last visited Mar. 20, 2018) ("I'd like to see a remotely operated plane perform a high-speed takeoff abort after an engine failure, followed by a brake fire and the evacuation of 250 passengers. I would like to see one troubleshoot a pneumatic problem requiring a diversion over mountainous terrain. I'd like to see it thread through a storm front over the middle of the ocean. . . . Hell, even the simplest things. . . . On any given flight, there are innumerable contingencies, large and small, requiring the attention and *visceral* appraisal of the crew.").

[130] *See* Appendix A ("One more explanation"), *infra* notes 166–167 and accompanying text.

judgment.[131] Nevertheless, the leap from being rational within one's own mind to acting on one's rational conclusion is still more a question within philosophy than within science. It likely involves the perception of meaning that Melanie Mitchell identified above.[132] It likely involves issues of consciousness and self-awareness.[133] It likely involves mysteries of determinism and free will. I find it more than a little ironic that one of Turing's suggestions for making a machine seem more like a human—that is, to appear to have free will—was to make it act randomly.

> [I]t is certain that a machine which is to imitate a brain must appear to behave as if it had free will, and it may well be asked how this is to be achieved. One possibility is to make its behavior depend on something like a roulette wheel or a supply of radium. The behaviour of these may perhaps be predictable, but if so, we do not know how to do the prediction.[134]

Turing perceived a connection between this randomness, free will, and the "aha" moment of the generation of new meaning: "We must not always expect to know what the computer is going to do. We should be pleased when the machine surprises us, in rather the same way as one is pleased when a pupil does something which he had not been explicitly taught to do."[135]

---

[131] *See, e.g.*, PENROSE, *SUPRA* NOTE 21, at 529–34. I like how Copeland and Sylvan summarized it in 1999. Who knows what the future will bring? As an example, a notional quantum Turing machine was described in 1985 and, by 1995, "pieces of experimental quantum hardware [were] sitting on laboratory benchtops." Copeland & Sylvan, *supra* note 98, at 64. Hence:

> [W]e would be profoundly surprised if the physics of the real world can be properly and fully set out without departing from the set of Turing-machine-computable functions. . . . In short it would—or should—be one of the greatest astonishments of science if the activity of Mother Nature were never to stray beyond the bounds of Turing-machine-computability.

Id.

See also B. Jack Copeland, *Turing's O-machines, Searle, Penrose and the brain*, 58 ANALYSIS 128 (1998).

Nevertheless, writing this essay has caused me to converse with some very accomplished people in the artificial intelligence community. I'm convinced, as a result, that whether you think AI can replicate the mind (or whether computational decidability matters to anything) is an issue of belief and affect rather than knowledge. I'm inclined to agree with Professor Byers (and Roger Penrose) that there is something non-algorithmic about creativity, but I'm open to demonstration that it isn't! *See supra* notes 91 and 124.

[132] *See supra* note 117 and accompanying text.

[133] MITCHELL, *supra* note 14, at 184. *See also* Dilip K. Prasad & Janusz A. Starzyk, *A Perspective on Machine Consciousness*, COGNITIVE 2010: 2D INT'L CONF. ON ADVANCED COGNITIVE TECHNOLOGIES & APPLICATIONS 109 (2010); Murat Aydede & Güven Güzeldere, *Consciousness, Intentionality and Intelligence: Some Foundational Issues for Artificial Intelligence*, 12 J. EXPERIMENTAL & THEORETICAL ARTIFICIAL INTELLIGENCE 263 (2000).

[134] Alan Turing, *Can Digital Computers Think?*, *in* THE ESSENTIAL TURING, *supra* note 1, at 482, 484.

[135] *Id.* at 485.

Deciding to decide on incomplete information is important. It is possible that an incomplete calculation, whether by human or automaton, provides useful information. Taking advantage of an incomplete calculation may well be better than flipping a coin or not thinking at all. Choosing to act on incomplete information is what Nobel Prize winner Herbert Simon called "satisficing."[136] A satisficer acts on information that is not perfect but merely good enough.[137] The human satisficer chooses to stop calculating and to act.[138] The human satisficer decides to stop worrying about whether the problem is decidable. A human can program a digital computer to be a satisficer, to make a decision that is merely good enough but not perfect, whether or not the problem is decidable. But the digital computer, on its own, even if equipped with neural networks or deep learning, cannot be assured that it can decide that every problem-solving program it runs is decidable. If it is running an undecidable program, it can't, on its own, satisfice. Acting before the calculation is complete and while the outcome is uncertain, deciding while the pinwheel is still spinning, satisficing rather than maximizing, is the least automaton-like and most human thing that a lawyer can do.[139]

Beyond that, choosing and acting on *any* information involves life commitments beyond mere thinking. I have previously written at length about the dilemma of the "two-handed" lawyer ("on one hand but on the other hand") who can analyze forever on either side of an issue, but cannot come to a conclusion. The thesis is that making judgments and deciding are matters of will (whether free or not) rather than rationality, and closer in nature to acting than to thinking.[140] To insist on the coming "singularity" of purely rational machine-thinking to ignore the theoretical constraints imposed by the mathematics of digital computers, and to put aside the issues of meaning, intention, and action all strike me as precisely those kinds of System 1 thinking errors about which Kahneman warns. The idea that algorithms will solve all the hard problems is, I suppose, comforting to some. I find it as odd a faith as the belief that a messiah will set everything right. It is simply another, and perhaps more sophisticated incarnation

[136] BARRY SCHWARTZ, THE PARADOX OF CHOICE: WHY LESS IS MORE 77–79 (2005).

[137] *Id.*

[138] *Id.*

[139] The philosopher Maurizio Matteuzzi, in assessing the computational limits of AI, suggests distinguishing between the deterministic procedures (i.e., the algorithms) by which either machine or human accomplishes a goal, and the non-algorithmic, non-deterministic freedom under which goals themselves are determined. *See* Maurizio Matteuzzi, *Why AI is Not a Science*, *in* FRANCHI & GÜZELDERE, MECHANICAL BODIES, *SUPRA* NOTE 84, AT 409, 420.

[140] *See* LIPSHAW, *supra* note 111, at 130–34; Lipshaw, *Dissecting, supra* note 111, at 268–72. Nevertheless, the philosophy of the relationship between thought and action is a subject for another time. For a challenging philosophical reconciliation of strictly logical, analytical, empirical, or naturalistic views of the world, on one hand, with how practical decisions to act get made, on the other, *see* MICHAEL THOMPSON, LIFE AND ACTION: ELEMENTARY STRUCTURES OF PRACTICE AND PRACTICAL THOUGHT (2008).

of what Kahneman calls the "narrative fallacy": "[o]ur comforting conviction that the world makes sense."[141] The allure of algorithms—rules on steroids—is that they are more efficient, they take less psychic (or real) energy than confronting, deciding, and acting under conditions of uncertainty.[142] The hope is that if we get them right, we can substitute our lazy rules of thumb for the even lazier, but significantly more efficient, comfort of letting something automated do the deciding.

III. Legal Education in an Automatonic World

If it isn't clear by now, the pinwheel has been spinning in my head for a long time. Pure thought and pure rationality as a means to figuring everything out and getting to ***THE ANSWER*** are enticing.[143] I could let the pinwheel spin for a long time. One of the things to which I have reconciled myself over a long business and academic career is that there are some fundamentally unresolvable paradoxes that might never get figured out, like completeness and consistency in arithmetic, or complementarity in quantum physics. I derive joy from playing with concepts, but there seems to me to be a real complementarity, a fundamental non-overlap, between merely thinking and acting. It's why the existentialists' critique of reason appeals to me. At some point, you have to stop thinking, accept the uncertainty and even absurdity of the world, and do something. It is the Halting Problem in action.

This, then, is the perspective from which I want to discuss the implications for lawyering and legal education arising from the theoretical constraints that the mathematics of digital computing impose on Kearse. If we were to rewrite the curriculum on an almost blank sheet of paper, knowing what we presently know about the theoretical constraints on a machine, what we would we do? What human lawyers will bring uniquely to the party, as long as the lawyer-automatons are digital, is the ability always, for every problem in what Reeke and Edelman called the unending compounding of exceptions to exceptions that are present in real life situations,[144] to say, "Enough is enough; I don't know whether the problem is ultimately solvable by more reasoning, I perceive a need for resolution, so I am deciding the time has come to make a decision and to act."

One of my first reactions to being a real lawyer and no longer merely a student was a Halting Problem. One of the primary tasks of a summer associate or first-year lawyer is to prepare research memoranda and briefs. For the first time, what I wrote had no page limits assigned by a professor; the work was done when *I*

---

[141] KAHNEMAN, *supra* note 110, at 201. For a mathematician's discussion of the affective aspect of the certainty of science, see WILLIAM BYERS, THE BLIND SPOT: SCIENCE AND THE CRISIS OF UNCERTAINTY 48–58 (2015).

[142] Barry Schwartz & Kenneth E. Sharpe, PRACTICAL WISDOM: ARISTOTLE MEETS POSITIVE PSYCHOLOGY, 2005 J. HAPPINESS STUD. 1, 8; BARRY SCHWARTZ & KENNETH SHARPE, PRACTICAL WISDOM: THE RIGHT WAY TO DO THE RIGHT THING 43–45 (2010).

[143] Hence, it isn't just "any" answer, but "the" answer in italics, capitals, and boldface.

[144] Reeke & Edelman, *supra* notes 89–91.

decided the work was done. On any particular issue the partner had assigned to me, I very well might not find a Michigan case, or anything helpful in the relevant treatises and ALRs. Was it worth a trip over to the bar association library to scroll manually through hundreds of headnotes in the West Decennial Digests?[145] How much of my own analysis should I apply to the problem? Nobody was going to be able to answer that question for me. Kearse would change that calculation, at least in regard to the trip to the library. But would Kearse know when to stop thinking about the problem if I didn't do something?

Enough with the spinning of the pinwheel. What should law school look like in a world of lawyer-automatons like Kearse? It should assume Kearse can do everything a lawyer can do, except there may be instances in which it doesn't know if the program will halt and therefore it needs to take action on an otherwise logically undecidable matter. If I were clever, I would call my curriculum something like law or lawyering in action, but a bunch of thoughtful people have already done that, and they are onto something relevant to my thesis here. In 1994, Stewart Macaulay captured the essence of it in a tribute to the University of Wisconsin Law School's particular scholarly tradition.

> [T]he Wisconsin approach is that law must be studied in its full social context.
>
> * * *
>
> Of course, law schools must teach some of the body of received wisdom and lawyer skills and teach this well. . . . But this is not enough. The challenge is to prepare students to deal with the law in action during their legal careers. Rules matter, but we cannot teach our students all the rules they might have to master to practice the day they are admitted to the bar.
>
> * * *
>
> At our best, Wisconsin professors have emphasized testing ideas in practice. To paraphrase Frank Remington, the criminal law becomes real in the front seat of a squad car.[146]

Law-in-action recognizes that nothing matters in that sacred corpus of subjects like contracts, secured transactions, banking law, federal income tax, or family law except to the extent lawyers and their clients use them to *do* something.[147] A

[145] I date myself here. Lexis and Westlaw first appeared while I was in law school. Pong was our computer game (although I recall a version of Battleship if you went to a particular computer terminal in a particular building near the law school at Stanford.) Our introduction to legal research was manual and via print editions of books, West Digests and Shepard's Citations.

[146] Stewart Macaulay, *Wisconsin's Legal Tradition*, 24 GARGOYLE 6, 9 (1994).

[147] In Macaulay's contract law textbook, the authors observe "there are large gaps between the law school law of contract, what happens in courts, and what practicing lawyers do" and "contract doctrine clearly is only one part of what lawyers need to understand to serve their clients." They rightly warn students against the expectation "that your professors are going to hand you a beautifully worked out, consistent, and

curriculum of action, one that recognizes the full extent of Kearse's capability, only now just being tapped in AI programs that can do things like write sophisticated private placement memoranda,[148] would focus on what Kearse is least able to do. I am going to limit my comments here to some general thoughts about the subjects we teach, and how and to whom we teach them.

As to how the curriculum divides up subject matter areas, the digital legal research engines that have been available for forty years, like Lexis and Westlaw, don't need them to do their work. Kearse certainly won't care whether a problem gets classified in the law school curriculum as falling under contract, tort, civil procedure, or patent law. Subjects are helpful heuristics, but there is nothing in nature or metaphysics that calls for them. Oliver Wendell Holmes famously observed of law generally, "It is revolting to have no better reason for a rule of law than that so it was laid down in the time of Henry IV."[149] Holmes should have been equally revolted by the evolution into eternal truth what were merely the helpful heuristics for dividing the received wisdom into digestible chunks. When I was practicing, seeing a problem as being within a legal subject was merely a shortcut for getting to something helpful in the doctrine. To be sure, I am not a complete anarchist, because one has to start the teaching somewhere. But every year, within the first couple weeks of Contracts I, I have to demonstrate to students that the result in *King v. Trustees of Boston University*,[150] purportedly a case on the *contract* doctrine relating to the elements of a promise, really turns on the *civil procedure* standard of appellate review of a jury verdict.

I see only two reasons to organize classes around the received wisdom of subject matter classifications in the law. The first is the bar examination. The National Conference of Bar Examiners promulgates a Uniform Bar Examination consisting of four parts.[151] The Multistate Bar Examination (MBE) is the six-hour, 200-question, multiple-choice exam. The MBE covers seven subject matters areas: Civil Procedure, Constitutional Law, Contracts, Criminal Law and Procedure, Evidence, Real Property, and Torts. Its administration has been uniform across the United States and its territories for many years (presently the only jurisdictions not adopting the MBE are the civil law jurisdictions of Louisiana

coherent system called 'contract law.'" Contract law is, instead, "a tool that you can use to try to solve your client's problems, rather than a set of answers to all your questions." STEWART MACAULAY ET AL., CONTRACTS: LAW IN ACTION 15–18 (3d ed. 2010).

[148] *See supra* note 6–7.

[149] Oliver Wendell Holmes, *The Path of the Law*, 110 HARV. L. REV. 991, 1001 (1997).

[150] King v. Trustees of Boston Univ., 647 N.E.2d 1196 (1995).

[151] NAT'L CONFERENCE OF BAR EXAMINERS, UNIFORM BAR EXAMINATION, http://www.ncbex.org/exams/ube/ (last visited Mar. 20, 2018). Twenty-eight jurisdictions have adopted the UBE, although some implementations have not yet occurred. For example, our state, Massachusetts, has adopted the UBE, but its first administration will be in July, 2018. MASS. BOARD OF BAR EXAMINERS, UNIFORM BAR EXAM, http://www.mass.gov/courts/court-info/sjc/attorneys-bar-applicants/bbe/ (last visited Mar. 20, 2018). In addition, the NCBE promulgates the Multistate Professional Responsibility Examination (MPRE). Every jurisdiction except Maryland, Puerto Rico, and Wisconsin requires it.

and Puerto Rico).[152] The uniform essay portion, the Multistate Essay Examination (MEE), has been adopted in thirty-eight jurisdictions. The MEE covers those subjects plus Business Associations, Conflict of Laws, Family Law, Trusts & Estates, and Secured Transactions (Article 9 of the UCC).[153]

Forty-five jurisdictions have also adopted two uniform "performance tests" called the Multistate Performance Test (MPT). The MPT asks the examinee to complete a legal assignment memorandum to a supervising attorney, a letter to a client, a persuasive memorandum or brief, a statement of facts, a contract provision, a will, a counseling plan, a proposal for settlement or agreement, a discovery plan, a witness examination plan, or a closing argument, using facts and legal source materials supplied as part of the examination.[154] The NCBE's description of the MPT is interesting: it is "designed to test fundamental lawyering skills in a realistic situation. It is not a test of substantive knowledge."[155]

The upshot is that, for the vast majority of law schools in the vast majority of jurisdictions, the only subjects that have an arguable claim for teaching in the traditional manner of received wisdom are the six MBE subjects, the six additional MEE subjects, and professional responsibility. The irony, of course, is that every subject in practice will take place in circumstances most resembling the MPT, even though the MPT purportedly tests "skills" and not substantive knowledge. My claim here is that Kearse might well be able to pass all of the sections of the bar examination *except* the MPT, because the MPT requires the examinee to decide when to halt.

The second reason to organize classes around traditional subject matter classifications is effective pedagogy. There is something to thinking like a lawyer, but I believe Kearse will end up being able to undertake much of it, to the extent it is deductive, inductive, or even associative (i.e., dependent on pattern recognition).[156] The courses we use to teach the core of "thinking like a lawyer" are arbitrary, but I see no reason not to use the traditional first-year subjects that happen to be those covered on the MBE: Contracts, Civil Procedure, Torts, Real Property, Criminal Law, and Criminal Procedure. Beyond those courses, I would scrap the casebooks and reorganize the pedagogy, in the current jargon, "to flip the classroom." I might have to accommodate subjects that get tested on the MEE, but that would be the only reason. Just as criminal law in action occurs in the front seat of a squad car, evidence law in action occurs in a court room, family law in action occurs when counseling a distraught spouse or facilitating a

[152] NAT'L CONFERENCE OF BAR EXAMINERS, MULTISTATE BAR EXAMINATION, http://www.ncbex.org/exams/mbe/ (last visited Mar. 20, 2018).
[153] *Id.*
[154] NAT'L CONFERENCE OF BAR EXAMINERS, MULTISTATE PERFORMANCE TEST, http://www.ncbex.org/exams/mpt/ (last visited Mar. 20, 2018).
[155] NAT'L CONFERENCE OF BAR EXAMINERS, NCBE EXAMS, http://www.ncbex.org (last visited Mar. 20, 2018).
[156] LIPSHAW, *supra* note 111, at 17.

surrogacy, estate law in action is planning a succession, and secured transactions in action is a project finance matter.[157]

Beyond the thirteen bar examination subjects, I cannot see any reason, other than inertia, why any area of substantive law should be taught other than as law-in-action.[158] That means treating the relevant substantive law or document as something a lawyer-human or lawyer-automaton can determine or create, and leaving for the lawyer-human the problem of deciding when enough legal thinking is enough (i.e., to halt). If I have a matter that involves the issuance of securities, for example, there will be plenty of resources that give me an overview of the area, a guide to the potential minefields, and details of the areas in which I need to concentrate. I will learn the "law" in connection with solving a problem. I don't think that view will be much of a surprise to any lawyer who has spent much time in practice. I sit on the board of a homeowners' association in Michigan. Because I'm the only lawyer around, I get called on to answer questions from time to time. One, for example, involved the question whether an owner could trim the intruding limbs of a tree whose trunk straddled the lot line. Another involved whether docks extending into the lake from owners' shores violated the riparian rights of adjacent owners. I never studied those subjects in law school, but I was able to come up with answers very quickly. When students take my course in Securities Regulation, the reality is that they are going to have to learn all sorts of securities related issues on their own in practice. I don't teach anything on Section 16 short-swing trading, causation in 10b-5 litigation, class certification under the Private Securities Litigation Reform Act, or many other sub-topics in the area.

Finally, while this may be an issue for "independent" law schools not affiliated with a broader university, I would take to heart Frank Remington's admonition about criminal law in action, recognizing that almost any law in action involves a non-lawyer, whether a client or another professional who is also engaged to solve the problem. I'm on record as suggesting that I would scrap the current "law student only" introduction to business entity law in favor of a co-taught interdisciplinary course for law and business students called "The Law and Finance of Business Organizations."[159] In that class, each professional-in-training would find the substantive knowledge outside the classroom (appropriately guided by the professor), and bring his or her expertise back to the classroom to bear on the problem to be resolved within it. What the participants would learn is the *meaning* their disciplines bring to the transaction. Lawyers would learn quickly that business imperatives take precedence over legal risk reduction, and business

[157] I don't have much experience with conflict of laws in action, but I expect it is part of a litigation practice.

[158] Looking at our own course descriptions at Suffolk, I can see the list would be long, beginning alphabetically with Administrative Law, Admiralty, Antitrust, Banking Law, and Basic Federal Income Tax, and concluding with Taxation of Intellectual Property, Trade Secret Law, Trademark Law, and Workers' Compensation.

[159] Jeff Lipshaw, *Crisis Management, AI and the United Fiasco 'With a Nod to Immanuel Kant'*, LEGALBUSINESSWORLD (Aug. 3, 2017), https://www.legalbusinessworld.com/single-post/2017/08/03/Crisis-Management-AI-and-the-United-Fiasco-'With-a-Nod-to-Immanuel-Kant'.

students would learn how to incorporate legal advice into a decision. All of them would decide together when it was time to decide. I'm a business lawyer and teacher, but the analogs would be that family lawyers learn alongside family counselors and psychologists, criminal lawyers alongside criminal justice professionals, estate lawyers alongside investment advisers, and so on. We would be focusing legal education on those areas Kahneman concedes are difficult for an algorithm-based automaton like Kearse: data inputs are idiosyncratic or hard to code, or the decision involves multiple dimensions or depends on a negotiated outcome.[160]

Conclusion

One of the luxuries of the "futurist" genre is being able to be provocative, whether about Singularities or Halting Problems, without much danger of being proved wrong. The test for the accuracy of my assertions about the theoretical constraints on AI lawyering is likely to occur when the only people left with any conceivable personal interest in my SSRN downloads or the number of citations to this essay are my as-yet unborn grandchildren.[161] I also recognize that inertia is real, faculties are conservative and slow to change, and that what I'm suggesting won't occur overnight. But if we are going to be futurists about artificial intelligence and the legal profession, it should be with clear and unfrenzied heads about the theoretical constraints on what digital (at least) lawyer-automatons like my Kearse will be able to do. To paraphrase Turing, whether we are defending humans or automatons, let us avoid well-chosen gibberish. Let us turn the center of gravity of human interest into philosophical questions of what in principle can be done. And then act accordingly.

[160] *See* Kahneman, *supra* note 119.
[161] I already know that my children are not interested.

Appendix A

This is a guide to working through Turing's proof in Section 8 of *On Computable Numbers.*

1. Turing's First Demonstration

Section 8 of *On Computable Numbers* begins with Turing's proof that some numbers are not computable, using the "diagonal" method. By definition, Turing machines can compute sequences of numbers starting with a blank tape. The sequence is "computable" if a circle-free machine can compute it. Some infinite number sequences are not computable. Consider the following set of sequences:

	Input I:	1 2 3 4 5 6 ... n(t)
Turing machine M	1	**1** 0 0 1 0 1 ...
" "	2	1 **0** 0 1 0 0 ...
" "	3	0 0 **0** 1 1 1 ...
" "	4	1 1 1 **1** 0 1 ...
" "	5	0 1 0 1 **0** 1 ...
" "	6	0 0 0 0 0 **0** ...
	.	
	.	
	.	
Turing Machine M	n	**1 0 0 1 0 0 ... 1**
Sequence β	t	**0 1 1 0 1 1 ... 0**

Each sequence continues infinitely off the page to the right, and the list continues infinitely down off the bottom of the page. Each sequence consists of a series of numbers computed by a Turing machine M running an input I. The first sequence gets computed by the Turing machine with the smallest possible description number ("D.N.") of any circle-free machine, and it runs every input number, n, from 1 to infinity. The second sequence gets computed by a circle-free Turing machine with the next smallest D.N. And so on for each such Turing machine, D.N. n, from 1 to infinity. Stretching to infinity both ways, this would enumerate every computable sequence from every possible Turing machine and every input. (In my grid, I have shown the output of each machine/input combination, M(n, n), as a binary digit. Those outputs are arbitrary and could stand for much longer numbers. I've expressed the description numbers for M and I in base ten to be clearer, but they could be binary numbers as well.)

Assume that the machine H is the algorithm that can determine whether another machine running on a particular input generates an output (i.e., halts, or is computable, or is circle-free). If the output of H running M on I = 1, M halts; if the output = 0, M does not halt. The output of H is the diagonal sequence β. It is constructed, by definition, by proceeding down diagonally (see the bold-faced digits), and switching each bolded figure from 0 to 1, or vice versa. Further assume that there is a number, t, representing the description number of H itself. If H can determine whether the output of any instance of M running I is computable, then the output in the diagonal for H itself, that is, the intersection of row t and column t, should be 1. But H(t, t) only equals 1 when M(t, t) equals 0. However, M running t on t cannot generate a 0 because sequence β is, by definition in the diagonal construction, not within the table of M running n on all n. Hence, H(t, t) should be 0. H(t,t) cannot be both 1 and 0. There is a contradiction and H cannot exist.[162]

2. Turing's Second Demonstration

Turing then provided a second demonstration that some numbers were not computable, this time using Turing machines that operate on themselves (what I call "P running on P"). What I call machine H is there called D. It is assumed to be a program (or machine) that can determine if any other machine is circle free (i.e., whether it will halt or loop).[163] H+ is the name of a machine which includes D. The first job of H+ is to compute a sequence, called β′, of numbers along the tape. β′ will consist of figures computed on the diagonal of a grid as created earlier. (Unlike β in the first demonstration, for β′ here we don't change the 0s to 1s and vice versa.) For this exercise, we assume that all the enumerable Turing machine description numbers, 1 to N, run down the vertical. And the inputs for those Turing machines, running along the horizontal will be the same Turing machine description numbers from 1 to N. When H+ is running, the output on the diagonal is the result of the computation when a Turing machine with description number, N, uses its own description number, N, as the input. (Each of the Turing machine numbers, N, represents a segment of the tape expressed in binary digits.)

Turing machine I (as Input):		1 2 3 4 5 6 … N
Turing machine M	1	**1** 0 0 1 0 1 …
“ “	2	1 **0** 0 1 0 0 …
“ “	3	0 0 **0** 1 1 1 …
“ “	4	1 1 1 **1** 0 1 …
“ “	5	0 1 0 1 **0** 1 …
“ “	6	0 0 0 0 0 **0** …
	.	
	.	
	.	
Turing Machine	N	
Sequence β'		**1 0 0 1 0 0** …

As H+ generates sequence β′, the program D (which is part of H+) determines whether each output in sequence β′, from the beginning to N-1, is "circle-free" (i.e., computable with that machine on that input). After N-1 has been tested, a certain number of the figures, R(N-1), in the sequence β′ will have

[162] Turing, *On Computable Numbers*, *supra* note 21, at 72; Copeland, *Computable Numbers*, *supra* note 1, at 33–35; PENROSE, *supra* note 21, at 75–83.

[163] Turing, *On Computable Numbers*, *supra* note 21, at 72–73.

been found to be description numbers of Turing machines that are circle-free. Then, in the N segment, D tests N. If N is circle-free, H+ will show that R(N) = 1 + R(N-1). At that point, H+ has calculated a sequence β′ with a description number, N, which includes R(N) circle-free, or computable numbers, within the sequence. If D determines N is not circle-free, then the number of circle-free description numbers remains R(N-1), and H+ moves on to segment N+1, which will be Turing machine N+1 running with input N+1. And so on.

By construction, at this point H+ is itself circle-free because D (which is part of H+) is assumed to work on each segment of the sequence. D decides whether the segment represents a circle-free description number in a finite number of steps. If N isn't circle-free, that segment is finished, and H+ moves on to test N+1. If N is circle-free, it means H+ can now calculate that there are R(N) circle-free segments in the tape. In either case, H+ finishes its job for that segment of the sequence β′ in a finite number of steps.[164]

We have reached the point in Turing's second demonstration where he has established that the hypothetical H+ machine is circle-free when creating sequence β′. To have H+ run on itself, he hypothesizes H+'s own description number, i.e., some N, and calls it *K*. The diagonal grid with K looks like this:

Turing machine I (as Input):		1 2 3 4 5 6 … K … N
Turing machine M	1	**1** 0 0 1 0 1 …
“ “	2	1 **0** 0 1 0 0 …
“ “	3	0 0 **0** 1 1 1 …
“ “	4	1 1 1 **1** 0 1 …
“ “	5	0 1 0 1 **0** 1 …
“ “	6	0 0 0 0 0 **0** …
	.	
	.	
	.	
	K	1 1 0 1 1 0 … **?**
	.	
	.	
	.	
Turing Machine	N	
Sequence β' at K		**1 0 0 1 0 0** … **?**

That is, when H+, running as before, reaches H+(*K, K*), it can't say that the machine represented by *K* (namely, H+!) is circular, because we've assumed by construction H+ is not. (In other words, if *K* weren't computable (i.e., it is circular), D would have done its job, and circle-free H+ ought to move on to test the next number in the sequence, *K*+1.)

[164] *Id.* at 73.

But in order for H+ to decide that *K* does halt (i.e., is circle-free), it would have to simulate what H+ does, i.e., go back to generate and test the entire diagonal sequence β′ from the first number to H+(*K*-1, *K*-1). That is fine up to *K*-1. What happens when it hits H+(*K*, *K*)? It has to start all over again from 1 through K because K represents a machine, H+, designed to generate and test all the numbers in the sequence up to and including K. (To make matters worse, H+ is doing all of this on what Turing calls the "E-squares" of the tape; i.e., the segment of the tape in which the machine does its background scratch pad work and then erases it. So, the spinning pinwheel metaphor is appropriate.)

Hence, H+, which is circle-free by definition, when working on itself, goes into an infinite loop and does not halt. H+, the halting program, can't be possible because it halts when it doesn't halt, and never halts when it does halt. D, the part of H+ that determines whether a machine is circle-free, cannot exist.[165]

3. One more explanation

I have worked it out intuitively another way that is consistent with both Mitchell's and Copeland's descriptions.[166] It involves assuming H+ is a human computer, working out the calculations with pencil and paper. This is also consistent with Turing's own dictum that "[t]he idea behind digital computers may be explained by saying that these machines are intended to carry out any operations which could be could done by a human computer."[167]

Go back to Figure 2 in the main text. Assume program P is running on input P (or any other input) and is showing a spinning pinwheel. Assume H+ is a human being watching the pinwheel spin and doing a calculation to determine when it will stop. H+ is going to watch and calculate until it has an answer and then get up and leave. If H+'s calculation is that P is in a loop, i.e., that the spinning pinwheel will never stop, then H+ has nothing left to do, and it gets up and leaves (i.e., halts). But if H+'s calculation presently says that P isn't in a loop, it only knows that P isn't in a loop now. So, H+ has to keep sitting there, watching and calculating, as long as the pinwheel is spinning, and asking the question whether P is in a loop. Hence, if H+ keeps saying P will halt, and the spinning pinwheel keeps spinning, H+ has to keep running because its work isn't done. H+ won't give an answer to the question whether P will halt, i.e., that the spinning pinwheel will stop, as long as P hasn't halted and the spinning pinwheel is spinning! Then imagine H+ sitting there do it to itself, that is, trying to figure out whether its own pinwheel will stop.

[165] *Id.* at 73; Copeland, *Computable Numbers*, *supra* note 1, at 37–39.

[166] Mitchell, *supra* note 14, at 67; Copeland, *Computable Numbers*, *supra* note 1, at 39–40.

[167] Turing, *Intelligence*, *supra* note 12, at 444.