Background to Turing Machines: Highlights of Infinity and Reasoning

We often associate bugs with computation, but many of our ideas of computation in fact arose from a desire to *avoid* bugs in mathmematical reasoning, particularly in reasoning about infinity.

Ancient Greece

Zeno's Paradoxes of Infinity[1]

The Greek notion of infinity was that of the unreachable infinity one would get just beyond counting up in the positive integers.

Motion is Impossible: An arrow that is shot can never reach its target. Zeno showed that it would need to complete an infinite number of tasks before it could do so: At whatever position it is there would be the task of getting halfway to the target that would precede hitting the target. Once it got halfway, it would first have to go 1/2 that distance etc. None of these halfway points would ever get to the target.

Achilles and the Tortoise Even though the Tortoise is a lot slower than Achilles (the world's fastest human), if the Tortoise gets a head start Achilles will never catch him: Before Achilles can catch the Tortoise, he must get to the Tortoise's starting position, but by then the Tortoise will have moved on and still be ahead of Achilles. This can be repeated infinitely without Achilles ever catching up to the Tortoise.

Euclidean Geometry

Euclid's *Elements* was the model of reasoning that set the standards for mathematics for millenia. Geometry was the main subject of study. Greeks associated numbers with lengths and the Euclidean algorithm for GCD was designed to find the nicest unit of measure so that one could tile a rectangle with squares.

Euclid's *Elements* had only 5 axioms or 'postulates' from which the other items were derived using (somewhat formal) reasoning.

17^{th}- and 18^{th}-Century Mathematics

Newton, Leibnitz and Calculus: Differential and integral calculus were based on the notion of *infinitesimals*, infinitely small changes in values. These were great for a vast number of applications (Newton's work that introduced it was *Principia* which used it in his formulation of the foundations of physics) but it was unclear what these infinitesimals really meant. Leibnitz himself actually tried to formulate strict rules of reasoning but never was able to include infinitesimals in his reasoning.

Fourier Series: Fourier showed the value of representing functions as the sum of infinite series of trigonometric functions with different periods. This was good for calculation and had great practical value.

[1] Tom Stoppard's play *Jumpers* has some entertaining connections to both paradoxes. Its main character is a philosophy professor (and gymastics coach) who has a pet Tortoise names Achilles and who does class demonstrations in logic that include a bow and arrow.

19^{th} Century Mathematics

Paradoxes of Series: It was observed that minipulation of infinite series sums had some tricky issues. For example, the following is a correct series for the natural logarithm of 2:

$$\ln 2 = 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 - 1/8 + \cdots .$$

But we can then do the following:

$$\begin{aligned}
\ln 2 =& 1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 + 1/7 - 1/8 + \cdots \\
=& (1 + 1/3 + 1/5 + 1/7 + \cdots) - (1/2 + 1/4 + 1/6 + 1/8 + \cdots) \\
& \text{by grouping positive and negative terms} \\
=& (1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + \cdots) - 2(1/2 + 1/4 + 1/6 + 1/8 + \cdots) \\
& \text{by adding a copy of the 2nd sum on the left and subtracting it again} \\
=& (1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + \cdots) - (1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6 + 1/7 + 1/8 + \\
& \text{by multiplying out the factor of 2} \\
=& 0 \qquad \text{since the two terms are the same.}
\end{aligned}$$

This is impossible since $\ln 2 \neq 0$. Cauchy showed that this kind of reasoning is incorrect because the series for $\ln 2$ is *conditionally convergent* - it can produce different values by taking it in a different order. It turns out that many of Fouriers original 'proofs' of correct Fourier series formulas used the same reasoning!

Non-Euclidean Geometry (1830) Bugs in mathematical reasoning had been a problem and even Euclidean geometry was removed from its pedestal somewhat. The 5th of Euclid's axioms, the *parallel postulate* that given any point not on a given line there is a unique line parallel to the given line that passes through the point, had always seemed a little less obvious than the others and mathematicians had unsuccessfully tried to derive it from the others. Bolyai and Lobachechevsky (and Gauss) independently showed that it was possible to have a consistent notion of geometry that did not satisfy the parallel postulate.

Boole (1854): *The Laws of Thought* was the foundation of formal (propositional) logic. Its range of application was quite limited, however.

Cantor (1875): Cardinality of Infinite Sets Cantor defined the notion of the cardinality of infinite sets using 1-1 correspondences. With two truly amazing and surprising arguments, he showed that

$$|\text{Integers}| = |\text{Rationals}| \neq |\text{Reals}|.$$

The proof that the integers and rationals had the same cardinality introduced a new technique that has come to be called *dovetailing*. The proof that the reals are not of the same cardinality as of the rationals introduced another new techniques that has come to be called *diagonalization*. Both later figure prominently in arguments of Godel and Turing.

Cantor had a lot of trouble getting acceptance in his time. Kronecker, one of the most influential mathematician of his day, railed against Cantor's work and Cantor himself and made it hard to publish his work.

(Kronecker was conservative in his mathematical tastes and was later quoted as say (roughly translated) that "God made the integers, the rest is the work of man".)

Frege (1879) Introduced the predicate logic quantifiers "for all" and "there exists" as part of formal logic.

Peano (1889) Axioms for Number Theory: Used predicate logic to describe a fixed set of axioms that described the natural numbers under addition and multiplication. The induction axioms were actually given by an infinite set of axioms, one for each formula that could be easily produced.

Formally, these axioms included the constant 0 and function symbols $+, \cdot$ and $s(x)$ where s stands for *successor* - for example 1 is shorthand for $s(0)$. The Peano axioms are:

- $\forall x\ \neg(s(x) = 0)$
- $\forall x\ \forall y\ ((s(x) = s(y)) \rightarrow (x = y))$
- $\forall x\ (x + 0 = x)$
- $\forall x\ \forall y\ (x + s(y) = s(x + y))$
- $\forall x\ (x \cdot 0 = 0)$
- $\forall x\ \forall y\ (x \cdot s(y) = x \cdot y + x)$
- and for every formula P using these symbols we have the induction axiom for P:
  $(P(0) \wedge \forall x\ (P(x) \rightarrow P(S(x)))) \rightarrow \forall x\ P(x)$

Frege (1893) *Grundlagen Der Arithmetik* was to be the fundamental description of mathematics based on predicate logic and set theory. Frege completed the first volume and in 1902 had nearly completed the second, which was due to go to press the following year.

20^{th} Century[2]

Russell's paradox (1902) Bertrand Russell realized that there was a fundamental problem with Frege's formulations. In particular, the notion of set theory was very general and included sets such as "the set of all sets that satisfy a property P". For example, one could have "the set of all infinite sets". Since there are an infinite number of sets that are infinite, the set of all infinite sets would be a member of itself. However, sets we are used to do not have this property. Russell sent Frege a letter pointing out that there would be a problem with this collection of sets of the usual sort:

The set of all sets that are not members of themselves

He pointed out that the question "Is *this* set a member of itself?" had no good answer. As a result, Frege stopped work and never published his second volume.

[2]For a fun take on the history, which takes some liberties with the history see the graphic novel "Logicomix: An Epic Search for Truth" by Doxiadis and Papadimitriou

Hilbert's 23 problems (1900) Meanwhile, at the Mathematical Congress in 1900, Hilbert, who was the leading mathematician of the day, listed 23 grand challenge problems for mathematics. Behind several of these was the notion that mathematical questions could be formalized and mechanized in a way that would eliminate errors. Two of them explicitly asked for mechanical procedures for classification.

Russell and Whitehead (1913) ***Principia Mathematica*** : The title was a take-off on Newton's Principia. This introduced *type theory* as a way of avoiding the paradoxes that plagued Frege's approach. This notion of type theory was the origin of the types that eventually made their way into modern programming languages. The methods were rigorous and very mechanical. A drawback was that it took a very long time to prove anything. (It was notorious that a proposition somewhat late in the book proved $1 + 1 = 2$, though it could have been proved much earlier.) It seemed clear that this was essentially the right way to do logic. How could one get more of mathematics using logic?

It took a while to get notions that formalized what was meant by functions computed by mechanical means.

Godel (1925) Godel introduced the *primitive recursive functions* as a model of computation. In our modern language there could be written as programs with a single looping construct with nested loops:

$$\text{PERFORM } n \text{ TIMES} \langle \text{LOOP BODY} \rangle$$

where n is a variable. Unlike modern computer programs, this has the property that at the time a loop is entered, we know exactly the number of times that the loop body is executed. In particular the computations of primitive recursive functions always halt.

Ackermann's function (1927) Ackermann came up with a function that was obviously intuitively computable but he could prove was not a primitive recursive function: We can define a closely related function that typically is given the name Ackermann's function, by $A(x) = f_x(x)$ where

$$f_0(x) = x + 1 \text{ and for every } x, k \text{ we have } f_{k+1}(x) = f_k \circ \cdots \circ f_k(x)$$

where the composition of f_k with itself is done x times. In particular, $f_1(x) = 2x$, $f_2(x) = 2^x x$, and

$$f_3(x) > 2^{2^{2^{2^{\cdots 2}}}}$$

where the tower of exponentials has height x. One can show that because it grows so quickly, computing $f_k(x)$ requires at least k nested loops of the sort allowed for primitive recursive functions and therefore $A(x) = f_x(x)$ cannot be computed with any program with a fixed number of nested loops. Note that the *inverse* of the Ackermann function, typically denoted by $\alpha(n)$, is incredibly slowly growing and so is constant for all practical purposes. The Union-Find data structure has near-linear running time $O(n\alpha(n))$.

Entscheidungsproblem (1928) Hilbert and Ackermann ask for a solution to the following problem: Given a (good) logical system (consisting of sound logical inference and a set of axioms and symbols), find a mechanical procedure that will determine whether or not a given statement is true in every world in which the axioms hold.

Godel's Completeness Theorem (1929) If a logical statement is valid then it can be proved via a finite deduction.

Godel's Incompleteness Theorems (1931) No consistent system of logic described by an effective (primitive recursive) procedure (i.e., an extension Peano's axioms of number theory) can prove all true statements about the natural numbers. In particular, there are logical statements about the natural numbers that cannot be proven true or false. (So, even the integers, which mathematicians like Kronecker felt were pure, presented unsolvable mathematical problems.) The key ideas used Cantor's notions of dovetailing for encoding and diagonalization to prove impossibility.

The problem of what the appropriate notion of computation should be was still open.

Turing, Church, Post, Kleene (1936) All four of these authors almost simultaneously developed what turned out to be equivalent notions of computation. (Church advised Kleene as a PhD student and Church also became Turing's PhD advisor after Turing's paper on the subject was largely written. Turing also spent time at Princeton with Godel.) The notions were

- **Turing Machines:** Turing, Post. See the handout with excerpts from their simultaneous papers.
- **λ-calculus:** Church. This is the basis of modern functional programming. The programming language LISP even explicitly includes `LAMBDA` in its language syntax to match the role in Church's system.
- **μ-recursive functions:** Kleene. This was a direct extension of Godel's primitive recursive functions with an extra operator.

So why do we focus on Turing? He did much more than the others in his formulation. He

- gave strong intuitive justification that his was the ultimate model of computation,
- developed the Universal Turing Machine (a Turing machine interpreter). This was the first language interpreter/compiler and treated programs as data. (The idea was inspired by Godel's work. Church's λ-calculus had somewhat similar ideas.) This became the basis for the stored program computer which Von Neumann later suggested.
- proved the Undecidability of the Halting Problem, which showed that no model that always halts (always produces total functions) can capture all of computation.
- proved that the Entscheidungsproblem is not solvable in finite terms. (Church did this also).
- proved, in an appendix, that his notion was equivalent to Church's.

Kleene also showed that his system was equivalent to Church's. Along the way, Church and Turing formulated their thesis that these models exactly mathematically capture what computations are possible.