Dawson Conway

The limitations of modern computers were known before they were even thought of. Computer programs were written decades before programmable computers. The basis for all computer science was actually created by mathematicians. Very few people understand the origins of the computer, but even fewer understand where the science behind them comes from.

In 1900, mathematician David Hilbert published twenty-three questions for mathematicians. Without three of those, computer science could not exist. Two of those questions called for complete and consistent systems of formal logic and predicate calculus. The last asked if it was possible to know if a program halted, or stopped, without running the program. Formal logic is just a way to work with true-false statements. Its operators are: and, or, not, if-then and if-and-only-if. An example would be “I have a cat; I have a dog; therefore, I have a cat and I have a dog (A, B ∴ A ∧B)”. Predicate calculus is more complicated, having variables and quantifiers. An example would be “there exists a cat such that the color of said cat is orange(x ∃ P(x)=y)”. The last question, the Halting Problem, asks for a test to see if a program runs forever, which ends up being the most important questions of computer science.

Kurt Gödel was a mathematician around the same time as Hilbert. He is famous for his completeness theorem and his incompleteness theorem, which answer questions one and two respectively. Completeness theorem says there is a complete and consistent theory of formal logic. Incompleteness theory ruins everything. All Gödel did was say “A = A cannot be proven.” If you prove A, you prove that it cannot be proven. If you disprove A, you prove that it can be proven. This flaw prevents a complete and consistent system of predicate calculus.

Alan Turing is considered by many to be the father of modern computers. He designed the first feasible stored-program computer, although his version was never built. He also built some of the first machines which preformed computations, the bombes used to decode German transmissions during World War One. His most important development, the one that defined computer science before computers were even thought of and answered Hilbert’s final question, was a purely hypothetical machine now called a Turing Machine. This device can perform any computation that any computer could ever compute, and it is simply a scanner, a printer, a “state” (like a stored variable), and an infinitely long tape. The machine reads the square under it, looks at its state, and based on those two things prints something and moves one square to the left or right. The most expensive high-end computer on the planet can’t do anything with data a Turning machine can’t do. The crazy thing is that Turing wasn’t trying to come up with a design for a computer; he was trying to model the human mind! The most important thing this Turing Machine did was to prove there was no program that answered Hilbert’s halting problem for all other programs.

This halting program is a big deal. Sometimes it is easy to tell if a program halts. If I told you to start at one and keep counting up until you got to zero, you would know that that would never happen. While this is easy for some programs, imagine being able to do it for any program, or, if questions one and two had worked out, for any true-false statement. That would mean that we could find out if any statement was true or false, without proving the statement. Simply testing to see whether the proofof statement has an end, or if it goes on forever, you would know whether that statement was provable, and therefore true. Even though Hilbert’s questions were proven false, there are still uses for them. Predicate Calculus still works just not always, and the halting program works for some programs, so in some instances this can be used to make discoveries and prove theorems. It is therefore extremely useful for converting data to knowledge.