Regarded by many as the father of modern computer science, Alan Turing provided an answer to the *Entscheidungsproblem*. This problem was devised by mathematician David Hilbert, and it asked if there was a way to output a "True" or "False" given a formal language and a logical statement in that language. Turing showed in a paper called *On Computable Numbers, with an Application to the Entscheidungsproblem* that there is no such solution to the *Entscheidungsproblem*.

In more formal terms, he showed that there is no general process for finding a function if the functional calculus K is provable. To show this, he starts off by constructing a function that depicts a computing machine, call it μ. Afterwards, he notes μ is undecidable, or $Un(\mu)$. He concludes by finding a general method for determining $Un(\mu)$ to be unprovable, and as a consequence, that there is a general method for showing whether M prints 0.

Turing proceeds in his paper by defining several functions that describe the configuration of a variable. He uses these functions to derive a formula for $Un(\mu)$. With substitution, he found that $Un(\mu)$ should be understood as "in some complete configuration of μ, S_1 appears on the tape." Following his finding, he sought to prove two crucial ideas; those being

- If S_1 appears on the tape in some complete configuration of μ, then $Un(\mu)$ is provable.
- If $Un(\mu)$ is provable, then S_1 appears on the tape in some complete configuration of μ.

These two ideas are lemmas, and they are used to help prove that the *Entscheidungsproblem* cannot be solved.

The first Lemma states: "If S_1 appears on the tape in some complete configuration of mu, then $Un(\mu)$ is provable. First, a configuration of a sequence of symbols is proposed and used to write a proposition, abbreviated CC_n. Keeping CC_n and $F^{(r)}$ in mind, an abbreviation of a formula previously discussed in the paper, the following is declared: $A(\mu)$ $F^{(n)}$ implies CC_n are provable.

Also abbreviated as CF_n and meaning the "n-th configuration of μ is X", where "X" represents the n-th complete configuration of mu. The provability of CF_n is possible. To exemplify, the finished configuration of symbols are all blanks, the m-configuration is q_1, and the scanned square is u for CF_0; therefore, the statement $A(\mu)$ implies CC_0 is nonessential.

The next part of the Lemma considers three cases regarding the movement of the machine when demonstrating that for each n, CF_n implies CF_{n+1} is provable. In any case, comparable arguments are made and formulas are implemented in a series of steps to validate the following: CF_n implies $CFn + 1$. Thus, CFn is provable for each n.

The last part of this lemma deals with the premise that S_1 will eventually be seen when mu is printed. When considering this and $A(\mu)$ and $F^{(n)}$ implies CC_n, stated above, and utilizing predicate calculus, we have proven that $Un(\mu)$.

Lemma two is a proof that if $Un(\mu)$ is provable then S_1 (i.e. 0) appears on the tape in some complete configuration of M. This lemma is essentially the converse of the first lemma, and is necessary in order to use an if and only if statement which is part of the final proof. The second Lemma in other words says that it is true that S1 appears somewhere on the tape of a complete configuration of M.

After the two lemmas have been proven, Turing states that he is "in a position to show that the *Entscheidungsproblem* cannot be solved". By Lemma one and two it can be said that the process for determining whether or not M ever prints a 0 is an impossible one. Therefore, the *Entscheidungsproblem* cannot ever truly be solved.

At the end of his paper is the appendix, which discusses computability and effective calculability. In it, Turing showed that every λ-definable sequence γ is computable. His proof of this makes use of the calculus of conversion and a "well-formed formula" N_n. First, he constructs a machine $\mathcal{L}$, supplied with some arbitrary formula M_γ, which writes the sequence γ. The way the machine $\mathcal{L}$ functions is divided into stages, where the n-th stage is devoted to finding the n-th figure of the sequence γ. He then shows that the first stage within the n-th part of the formation $\{M_\gamma\}(N_n)$. As a consequence, the formula is supplied to machine $\mathcal{L}_2$. This converts the machine successively into other formulae.

The proof wraps up from this point on. Turing finds a formula M_γ such that for every integer n,

$$\{M_\gamma\}(N_n) conv N_{1+\phi_\gamma(n)}.$$

To find the formula, Turing starts off by taking a machine μ that computers γ, and let's it take the complete configuration of μ through numbers. Then, he chooses a function, say $\zeta(n)$, to be the description number of the n-th complete configuration of μ, and gives the relation between $\zeta(n+1)$ and $\zeta(n)$ in the form of an equation like so:

$$\zeta(n+1) = \rho_\gamma(\zeta(n)).$$

Further, he uses his findings to show that a formula, call it V, helps create a piecewise function that either converts W.F.F.'s N_1, N_2, or N_3. He concludes the proof by showing that the formula M_γ is defined as desired above. Thus, showing that every λ-definable sequence γ is computable.

As a result, Turing's halting problem solution generalized Godel's incompleteness theorem. Why? The idea of Turing's halting problem says that given a function $T(m, x)$, a Turing machine m may or may not halt on input x. This idea corresponds to Godel's incompleteness theorem because Godel's theorem asserts that something is not provable. Similarly, the halting problem says that the algorithm searching for a proof of something doesn't halt. Thus, given Godel's theorem, it is not provable.