**Computability**

The typical method of proving a problem to be undecidable is with the technique of reduction. To do this, it is sufficient to show that if a solution to the new problem were found, it could be used to decide an undecidable problem by transforming instances of the undecidable problem into instances of the new problem.

Since we already know that no method can decide the old problem, no method can decide the new problem either. Often the new problem is reduced to solving the halting problem. (The same technique is used to demonstrate that a problem is NP complete, only in this case, rather than demonstrating that there is no solution, it demonstrates there is no polynomial time solution, assuming P ≠ NP.)

For example, one such consequence of the halting problem's undecidability is that there cannot be a general algorithm that decides whether a given statement about natural numbers is true or false. The reason for this is that the proposition stating that a certain program will halt given a certain input can be converted into an equivalent statement about natural numbers. If we had an algorithm that could find the truth value of every statement about natural numbers, it could certainly find the truth value of this one; but that would determine whether the original program halts, which is impossible, since the halting problem is undecidable.

Rice's theorem generalizes the theorem that the halting problem is unsolvable. It states that for any non-trivial property, there is no general decision procedure that, for all programs, decides whether the partial function implemented by the input program has that property. (A partial function is a function which may not always produce a result, and so is used to model programs, which can either produce results or fail to halt.) For example, the property "halt for the input 0" is undecidable. Here, "non-trivial" means that the set of partial functions that satisfy the property is neither the empty set nor the set of all partial functions. For example, "halts or fails to halt on input 0" is clearly true of all partial functions, so it is a trivial property, and can be decided by an algorithm that simply reports "true." Also, this theorem holds only for properties of the partial function implemented by the program; Rice's Theorem does not apply to properties of the program itself. For example, "halt on input 0 within 100 steps" is not a property of the partial function that is implemented by the program—it is a property of the program implementing the partial function and is very much decidable.

Gregory Chaitin has defined a halting probability, represented by the symbol Ω, a type of real number that informally is said to represent the probability that a randomly produced program halts. These numbers have the same Turing degree as the halting problem. It is a normal and transcendental number which can be defined but cannot be completely computed. This means one can prove that there is no algorithm which produces the digits of Ω, although its first few digits can be calculated in simple cases.

While Turing's proof shows that there can be no general method or algorithm to determine whether algorithms halt, individual instances of that problem may very well be susceptible to attack. Given a specific algorithm, one can often show that it must halt for any input, and in fact computer scientists often do just that as part of a correctness proof. But each proof has to be developed specifically for the algorithm at hand; there is no mechanical, general way to determine whether algorithms on a Turing machine halt. However, there are some heuristics that can be used in an automated fashion to attempt to construct a proof, which succeed frequently on typical programs. This field of research is known as automated termination analysis.

Since the negative answer to the halting problem shows that there are problems that cannot be solved by a Turing machine, the Church–Turing thesis limits what can be accomplished by any machine that implements effective methods. However, not all machines conceivable to human imagination are subject to the Church–Turing thesis (e.g. oracle machines). It is an open question whether there can be actual deterministic physical processes that, in the long run, elude simulation by a Turing machine, and in particular whether any such hypothetical process could usefully be harnessed in the form of a calculating machine (a hypercomputer) that could solve the halting problem for a Turing machine amongst other things. It is also an open question whether any such unknown physical processes are involved in the working of the human brain, and whether humans can solve the halting problem