Gödel's Theorems Quickly *

Ryan Brill

September 2021

Definition 1. *An* ***axiom*** *is a statement or proposition which is regarded as being established, accepted, or self-evidently true. An* ***axiomatic system*** *is any set of axioms from which some or all axioms can be used in conjunction to logically derive theorems (conclusions). A* ***theory*** *is a self-contained body of knowledge which contains an axiomatic system and all its derived theorems (conclusions). An axiomatic system is* ***consistent*** *if it lacks contradiction - that is, if it is impossible to derive both a statement and its negation from the system's axioms.*

Theorem 1 (Gödel's Completeness Theorem)**.** *If a set of axioms are consistent, then it is impossible to derive a contradiction using these rules. Conversely, if a set of axioms are inconsistent, then the inconsistency can be proved using these rules alone.*

Proof. We simply cook up objects to order as the axioms request them! And if we ever run into an inconsistency. that can only be because there was an inconsistency in the original axioms. □

Theorem 2 (Principle of Explosion)**.** *Any statement can be proven from a contradiction.*

Proof. Suppose the contradiction $P \wedge \neg P$ holds. Let Q be any proposition. Since P holds, $P \wedge Q$ holds. Since $\neg P$ holds and $P \wedge Q$, Q holds. □

Due to the Principle of Explosion, consistency is a key requirement for most axiomatic systems.

Definition 2. *The* ***Halting Problem*** *is: given a computer program Q, decide if Q ever halts.*

Theorem 3. *There is no computer program that solves the halting problem.*

Proof. Suppose, for contradiction, that such a program P exists, which, given another program A, decides whether A halts or not. Then we can use P to produce a new program P' that does the following. Given another program Q as input, P'

*Taken from *Quantum Computing Since Democritus* by Scott Aaronson. Great book! I wrote this to assist myself in explaining these concepts to my friends.

1. runs forever if Q halts, or
2. halts if Q runs forever.

Now, feed P' its own code as input. If P' halts, then it runs forever, and if it runs forever, then it halts. Contradiction! Therefore, P' – and by implication, P - can't have existed in the first place. □

Theorem 4 (Gödel's First Incompleteness Theorem)**.** *Given any consistent, computable set of axioms, there's a true statement about the integers that can never be proved from those axioms.*

If we didn't have the computability requirement, then we could simply take our "axioms" to consist of all true statements about the integers! In practice, this isn't a very useful set of axioms.

Proof. Suppose, for contradiction, that the First Incompleteness Theorem is false - that is, there exists a consistent, computable proof system F from which any statement about the integers could be either proved or disproved. Then given a computer program, we could simply search through every possible proof in F, until we found a proof that the program halts or a proof that it doesn't halt. This is possible because the statement that a particular computer program halts is ultimately just a statement about integers, because computer programs are merely bits. But this would give us an algorithm to solve the halting problem, which we know is impossible. Therefore, F can't exist. □

Theorem 5 (Gödel's Second Incompleteness Theorem)**.** *If a proof system F is consistent, then F can't prove its own consistency.*

Proof. Let P be a program that, given as input another program Q, tries to decide whether Q halts by searching through every possible proof and disproof that Q halts in some formal system F. Then we can use P to produce a new program P' that

1. runs forever if Q is *proved* to halt, or
2. halts if Q is *proved* to run forever.

Now, feed P' its own code as input. If P' finds a proof that P' halts, then P' will run forever, which is a contradiction. Also, if P' finds a proof that P' runs forever, then it will halt, which is a contradiction. So, P' must run forever without ever discovering a proof or disproof that it halts. But then, if F is consistent *and* can *prove* that F is consistent, then the previous argument becomes a *proof* that P' runs forever, a proof that should've been discovered in the search, and so is a contradiction! Hence if F is consistent, then it must be the case that F cannot prove its own consistency. □

Hence, the only mathematical theories pompous enough to prove their own consistency are the ones that don't have any consistency to brag about! If we want to prove that a consistent theory F is consistent, then we can only do it within a *more powerful theory* - a trivial example being $F + Con(F)$ (the theory F plus the axiom that F is consistent). But then how do we know that $F + Con(F)$ is itself consistent? Well, we could only prove that in a stronger theory: $F + Con(F) + Con(F + Con(F))$. And so on infinitely.

Theorem 6 (Löb's Theorem)**.** *If a theory F proves "if F proves X, then X", then F proves X.*

Proof. Assume that F proves "if F proves X, then X". Then F proves the contrapositive: "if X is false, then F does not prove X." In other words, we can prove in $F + not(X)$ that "$not(X)$ is perfectly consistent with F." Then $F + not(X)$ proves its own consistency, so by Gödel's Second Incompleteness Theorem, $F + not(X)$ must be inconsistent. Now, if F is inconsistent, then by the Principle of Explosion, F proves X. Also, if F is consistent, then because $F + not(X)$ is inconsistent, we must have that F proves X. Either way, F proves X. □

Corollary 7. *If X is not provable in F, then "if F proves X, then X" is not provable in F.*

Proof. Contrapositive. □

References

[1] Scott Aaronson (2013) *Quantum Computing Since Democritus*, Cambridge University Press.