Notes on Unprovable Theorems

We prove that in any "interesting" formalization of mathematics:

- There are true theorems that cannot be proved.
- The problem of checking whether a given theorem has a proof is undecidable.

The first part was proved by Gödel in 1931, the second part was proved by Church and Turing in 1936. The Church-Turing result also gives a different way of proving Gödel's result, and we will present this proof instead of Gödel's original one.

By "formalization of mathematics" we mean the description of a formal language to write mathematical statements, a precise definition of what it means for a statement to be true, and a precise definition of what is a valid proof for a given statement. A formal system is *sound* if no false statement has a valid proof. We will only consider sound formal systems. A formal system is *complete* if every true statement has a valid proof; we will argue that no interesting formal system can be sound and complete.

We say that a formalization of mathematics is "interesting" if it has the following properties:

1. *Mathematical statements that can be precisely described in English should be expressible in the system.*

   In particular, we will assume that, given a Turing machine M and an input w, it is possible to construct a statement $S_{M,w}$ in the system that means "the Turing machine M accepts the input w". We will also assume that the translation of $\langle M\rangle, w$ into $S_{M,w}$ can be performed algorithmically.

2. *Proofs should be detailed and convincing, and they should be easy to check step by step.*

   In particular, we will assume that given a statement S and an alleged proof, it is decidable whether P is a valid proof for S in the system.

3. *Simple proofs that can be precisely described in English should be expressible in the system.*

   In particular, if $S_{M,w}$ is the statement that says that Turing machine M accepts string w, and if indeed M accepts w, then we will assume that there is a proof of this fact in the system.

Perhaps we should justify these assumptions. If a system does not even allow you to formalize interesting theorems, then its limitations are clear from the start. In fact, one can come up with formal systems where are all true statements that you can write down are provable, but this is just because the system prevents you from writing down plenty of interesting statements. An example of such a system is the first-order theory of integers with addition discussed in Sipser's book. From our perspective, as computer scientists, a formalization of mathematics that does not allow us to talk about algorithms and Turing machines is not useful. One may still object that important subjects of mathematics (for example calculus and number theory) could be captured by a formal system that would still be intuitively interesting but that may not allow a definition of Turing machine. As explained in Sipser's book, however, already the first-order theory of integers with addition and multiplication satisfies our first assumption (using a clever encoding of Turing machines as integers).

Regarding the second assumption, in all formalizations of mathematics proposed so far, valid proofs have a very simple structure, and can be described by a simple grammar. In any such formalization, one can construct a program to decide whether a proof is valid using yacc and a few hours of spare time. In general, a definition of "proof" that makes it undecidable to check the validity of a proof contradicts our intuitive notion that a proof is something that convinces us that a statement is true. If we allowed undecidable proofs, then it would be easy to come up with formalizations where all statements are provable: just allow the string "it's trivial" to be a valid proof for true statements.

The third assumption will not be used in the proof of Gödel theorem. Indeed, if you have a system where you cannot prove that a given Turing machine halts, then you clearly have a system where some true statements don't have valid proofs. We will use the third assumption to prove the Church-Turing result that provability is an undecidable problem. This condition is there because you could have systems where the provability question is decidable for very trivial reason, for example because the system forbids proofs longer than 100 steps.

Theorem 1 (Gödel) *In every formalization of mathematics that satisfies the assumptions (1) and (2) there are true statements that have no proof.*

Proof: Suppose towards a contradiction that there is a formal system that satisfies assumptions (1) and (2) and in which all true statements have a proof. We will show that this implies that the Acceptance problem is decidable, which is impossible.

Let $(\langle M\rangle, w)$ be an input for the Acceptance problem. We construct the statement $S_{\langle M\rangle,w}$, which is true if and only if M accepts w, and the statement $(\text{not}S_{\langle M\rangle,w})$, which is true if and only if M does not accept w. Then we enumerate all strings P in lexicographic order, and for each of them we check if P is a proof that $S_{\langle M\rangle,w}$ is true and if P is a proof that $(\text{not}S_{\langle M\rangle,w})$ is true. Clearly, one of the two statements is true, and, by our assumption, it has a proof. So eventually we will find such a proof, and we will accept if we find a proof of $S_{\langle M\rangle,w}$ and reject if we find a proof of $(\text{not}S_{\langle M\rangle,w})$. Thus we have proved that the Acceptance problem is decidable and we have reached a contradiction. $\Box$

Theorem 2 (Church-Turing) *In every formalization $\mathcal{F}$ of mathematics that satisfies the assumptions (1), (2) and (3), the following language is undecidable:*

$$\text{Provability}_{\mathcal{F}} = \{S : \textit{ there is either a proof that } S \textit{ is true or a proof that (not } S\textit{) is true } \}$$

Proof: Suppose that for a formal system $\mathcal{F}$ that satisfies assumptions (1), (2) and (3) the language $\text{Provability}_{\mathcal{F}}$ were decidable. We show that this implies that the Acceptance problem would also be decidable.

Given an input $(\langle M\rangle, w)$ for the Acceptance problem, we consider the statement $S_{\langle M\rangle,w}$, and we run the algorithm for $\text{Provability}_{\mathcal{F}}$ on input $S_{\langle M\rangle,w}$.

If the algorithm accepts $S_{\langle M\rangle,w}$, it means that there is a proof of either $S_{\langle M\rangle,w}$ or its negation. Then we enumerate all strings P in lexicographic order until we find a valid proof for either of the statements. If we find a proof for $S_{\langle M\rangle,w}$ we accept, if we find a proof for (not $S_{\langle M\rangle,w}$) we reject.

If the algorithm for $\text{Provability}_{\mathcal{F}}$ on input $S_{\langle M\rangle,w}$ rejects, then we reject. This is because, if M accepts w, then, by assumption (3) there is a proof in the system that $S_{\langle M\rangle,w}$ is true. This means that the only unprovable statements of the form $S_{\langle M\rangle,w}$ are those in which M does not accept w. $\Box$