23 Unsolvable Problems in Logic

The fact that Peano Arithmetic is expressive enough to represent all computable functions means that some of the unsolvable problems from the theory of computation carry over to first-order logic. We know that the theory of computation is undecidable, that is we can formulate problems about computable functions that cannot be decided by an algorithm in a uniform way. Since we have shown how to represent computable functions in logic, the same problems can be formulated in first-order logic as well, which means that first-order logic must be undecidable. We are now in a position to give a unified account of this and two other negative results of logic:

- *Church's Theorem:* the undecidability of logic
- *Tarski's Theorem:* the undefinability of truth
- *Gödel's Theorem:* the incompleteness of systems of arithmetic.

23.1 Gödel Numberings and Diagonalization

The key to all these results is an ingenious discovery made by Gödel in the 1930's: it is possible to effectively enumerate all computable functions in a uniform way (via so-called *Gödel numberings*, and it is possible to define computable functions by *diagonalization* through the table of computable functions. Assumptions about decidability and computability then enable us to define a new computable function that is different from all other computable functions and thus leads to a contradiction. We will briefly sketch both methods.

Definition: A *Gödel numbering* is a mapping from a set of expressions to $\mathbb{N}$ that satisfies the following conditions

1. Different expressions receive different *Gödel numbers*. (*injective*)
2. The Gödel number of an expression can be effectively calculated. (*computable*)
3. It is effectively decidable whether a given number is a Gödel number or not.

It is easy to see that *the language of Peano-Arithmetic has a Gödel-numbering*. As we only use *finitely many different symbols* to write down arithmetic formulas we simply assign a number to every symbol. For instance, we may assign 0 to `(`, 1 to `)`, 2 to $\forall$, 3 to $\exists$, etc. Thus every formula corresponds to a unique finite sequence of numbers that represents the string of symbols needed to write down the formula.

There are multiple ways to *encode sequences of numbers* by natural numbers. One possibility is using polynomials as described in the representation of primitive recursion. Another one is extending the bijective encoding $\langle\rangle$ of pairs of numbers to an encoding $\langle\rangle^3$ of triples, quadruples, etc, and then to a bijective encoding $\langle\langle\rangle\rangle$ of arbitrary lists of numbers as follows:

$$\langle x\rangle^1 \equiv x, \quad \langle x_1, ..., x_{k+1}\rangle^{k+1} \equiv \langle\langle x_1, ..., x_k\rangle^k, x_{k+1}\rangle, \quad \langle\langle x_1, ..., x_n\rangle\rangle \equiv \langle n, \langle x_1, ..., x_n\rangle^n\rangle$$

All these functions are computable and have computable inverses, which means that given the assignments of numbers to symbols (and the rules for parsing formulas) we can determine whether a number corresponds to a formula and we are able to construct the formula if this is the case.

Diagonalization is a method for deriving contradictions from certain assumptions, which is best illustrated when considering an infinite set of functions. We construct a new function by going diagonally through the table of all function values and modifying the entry at each diagonal point. As a result we get a function that cannot be represented by a row of the table. The assumptions are usually crucial for proving that this diagonal function exists at all.

	0	1	2	3	4	5	6	...
f_0	$\boxed{f_0(0)}$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$	$f_0(5)$	$f_0(6)$	...
f_1	$f_1(0)$	$\boxed{f_1(1)}$	$f_1(2)$	$f_1(3)$	$f_1(4)$	$f_1(5)$	$f_1(6)$	...
f_2	$f_2(0)$	$f_2(1)$	$\boxed{f_2(2)}$	$f_2(3)$	$f_2(4)$	$f_2(5)$	$f_2(6)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$\boxed{f_3(3)}$	$f_3(4)$	$f_3(5)$	$f_3(6)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)$	$\boxed{f_3(4)}$	$f_3(5)$	$f_4(6)$	...
f_5	$f_5(0)$	$f_5(1)$	$f_5(2)$	$f_5(3)$	$f_5(4)$	$\boxed{f_5(5)}$	$f_5(6)$	...
f_6	$f_6(0)$	$f_6(1)$	$f_6(2)$	$f_6(3)$	$f_6(4)$	$f_6(5)$	$\boxed{f_6(6)}$	...
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Cantor's famous theorem about the nondenumerability of the set of total functions from $\mathbb{N}$ to $\mathbb{N}$ (which is equivalent to the nondenumerability of the set of real numbers), for instance, uses this argument. If we assume that all functions from $\mathbb{N}$ to $\mathbb{N}$ are denumerable, then we can define a new function $f{:}\mathbb{N}{\to}\mathbb{N}$ as $f(i) := f_i(i) + 1$. Because of the assumption, f must be some f_j in the table and we get $f(j) = f_j(j) + 1 = f(j) + 1$, a clear contradiction. As a result, the set of functions from $\mathbb{N}$ to $\mathbb{N}$ must be nondenumerable.

Gödel's important modification to that argument was the insight that diagonalization on computable functions is computable, provided we use a Gödel-numbering of computable functions. Gödel originally expressed his construction as an encoding of *self-reference* in formulas.

Definition: Let $\lceil \mathrm{X} \rceil$ denote the representation of the Gödel number of the formula X (in which `x` may occur free). The *diagonalization of X* is the formula $(\exists \mathtt{x})(\mathtt{x}{=}\lceil X \rceil \ \land\ X)$.

Lemma 1: Diagonalization is computable, that is there is a recursive function *diag* such that *diag(n* is the Gödel number of the diagonalization of X whenever n is the Gödel number of the formula X.

Proof sketch: Given a number n we can effectively determine whether it is a Gödel number of some formula X. Once we have X we can construct the formula $(\exists \mathtt{x})(\mathtt{x}{=}\underline{\boldsymbol{n}} \ \land\ X)$ and compute its Gödel number.

The difficult part of the actual proof is recasting the argument so that it deals with natural numbers only. One needs a specific Gödel-numbering for this purpose.

Diagonal Lemma: If $\mathcal{T}$ is a theory in which *diag* is representable, then for any formula $B(x)$ with exactly one free variable x there is a formula G such that $\models_T G \Leftrightarrow B(\lceil G \rceil)$.

Proof: Assume that $D(x,y)$ represents *diag* in $\mathcal{T}$ and let n be the Gödel number of the formula $F \equiv (\exists \mathtt{y})(D(x,\mathtt{y}) \wedge B(\mathtt{y}))$.

Define $G \equiv (\exists \mathtt{x})(\mathtt{x}=\underline{\boldsymbol{n}} \wedge (\exists \mathtt{y})(D(\mathtt{x},\mathtt{y}) \wedge B(\mathtt{y}))$ and let g be the Gödel number of G. Then G is logically equivalent to $(\exists \mathtt{y})(D(\underline{\boldsymbol{n}},\mathtt{y}) \wedge B(\mathtt{y}))$ and, by construction, the diagonalization of F, thus *diag*$(n)= g$.

Furthermore, since $D(x,y)$ represents *diag*, we know that $D(\underline{\boldsymbol{n}},\underline{\boldsymbol{g}})$ is valid in $\mathcal{T}$. Thus G is logically equivalent to $B(\underline{\boldsymbol{g}})$ and, since $\underline{\boldsymbol{g}}=\lceil G \rceil$, we have shown $\models_T G \Leftrightarrow B(\lceil G \rceil)$. □

Note that the diagonal lemma holds for Peano Arithmetic, as *diag* is representable in any theory that can represent the computable functions.

23.2 Incompleteness Results

The diagonal lemma shows that in theories that can represent computability all formulas have a fixed point. Fixed point constructors, on the other hand, lead to inconsistencies, as they make it possible to define formulas that are equivalent to their own negation. Before we prove this, let us introduce some useful notation.

Definition: Let $\mathcal{T}$ be a theory

1. $\mathcal{T}$ is called *consistent*, if there is no theorem in $\mathcal{T}$ whose negation is also in $\mathcal{T}$.
2. $\mathcal{T}$ is called *complete*, if for every formula X in the language of $\mathcal{T}$ either X or $\sim X$ is a theorem in $\mathcal{T}$.
3. A set S of formulas is called *decidable* if the set of Gödel numbers of S is decidable, i.e. if the characteristic function of that set is computable.
4. $\mathcal{T}$ is called *axiomatizable*, if there is a decidable subset of $\mathcal{T}$ whose logical consequences are exactly the theorems of $\mathcal{T}$. $\mathcal{T}$ is *finitely axiomatizable* if it is axiomatizable with a finite set of axioms.
5. A set $S \subseteq \mathbb{N}$ is called *definable in $\mathcal{T}$* if there is unary predicate R_S in the formal language of $\mathcal{T}$, such that for all $y \in \mathbb{N}$: $y \in S$ implies $\models_T R_S(\underline{\boldsymbol{y}})$ and $y \notin S$ implies $\models_T \sim R_S(\underline{\boldsymbol{y}})$. Definability extends naturally from sets (i.e. unary relations) to n-ary relations.
6. A theory $\mathcal{T}'$ is an *extension of $\mathcal{T}$* if $\mathcal{T} \subseteq \mathcal{T}'$, i.e. if every theorem in $\mathcal{T}$ is also one in $\mathcal{T}'$.
7. The theory of *arithmetic* is the theory over the language $\mathcal{L}(=,+,*,\mathsf{0},\mathsf{1})$ whose theorems are the formulas that are true in the *standard interpretation* $\langle \mathbb{N}, =, +, *, 0, 1 \rangle$.

Note that a set *S is definable in $\mathcal{T}$ if its characteristic function f_S is representable in $\mathcal{T}$*. For if $R_f(x,y)$ represents f_S, then $n \in S$ implies $f_S(n) = 1$ and $\models_T R_f(\underline{\boldsymbol{n}},\underline{\mathbf{1}})$, and $n \notin S$ implies $f_S(n) = 0 \neq 1$ and $\models_T \sim R_f(\underline{\boldsymbol{n}},\underline{\mathbf{1}})$. So the predicate $R_S(x) \equiv R_f(x,\underline{\mathbf{1}})$ defines S.

An immediate consequence of this insight is that *every decidable set of numbers is definable in Peano arithmetic* and in any other theory that can represent the computable functions, as the characteristic function of every decidable set is computable.

Representability and definability is preserved under extensions. In fact, if f is representable in $\mathcal{T}$ and $\mathcal{T}'$ extends $\mathcal{T}$ then f is representable in $\mathcal{T}'$ by the same predicate. Arithmetic is an extension of Peano Arithmetic, since the Peano axioms are true in both standard and non-standard interpretations, so representability and definability carries over from Peano arithmetic to arithmetic. However, there are certain functions and sets that cannot be represented anymore once a theory is expressive enough to represent the computable functions. A key observation, for instance, is that such *a theory cannot define its own theorems* by a formula.

Lemma 2: If $\mathcal{T}$ is a consistent theory in which *diag* is representable, then the set of Gödel numbers of theorems in $\mathcal{T}$ is not definable in $\mathcal{T}$.

Proof: Assume that $GN(y)$ defines the set of Gödel numbers of theorems in $\mathcal{T}$. By the diagonal lemma, there must be a formula G such that $\models_T G \Leftrightarrow \sim GN(\lceil G \rceil)$.

We show that both G and $\sim G$ are valid in $\mathcal{T}$, which contradicts the consistency of $\mathcal{T}$.

Assume G is not valid in $\mathcal{T}$. Then $\models_T \sim GN(\lceil G \rceil)$ by definition of GN and thus $\models_T G$ because of the above equivalence. This, in turn, means that G is a theorem in $\mathcal{T}$. Thus $\models_T GN(\lceil G \rceil)$ by definition of GN. Because of the above equivalence $\models_T \sim G$ must hold. □

The proof of Lemma 2 mimics in logic what Cantor's argument did to functions on natural numbers. We modify the predicate $GN(y)$ that we assumed to be definable ($\hat{=} f_i(i)$) into $\sim GN(y)$ ($\hat{=} f(i) = f_i(i) + 1$), determine its fixed point G ($\hat{=}$ the index j of f), and derive a contradiction $G \Leftrightarrow \sim G$ for that fixed point ($\hat{=} f(j) = f(j) + 1$).

An immediate consequence of Lemma 2 is that it is impossible to represent the arithmetically true statements by a formula in arithmetic.

Theorem: [Tarski's Indefinability Theorem]

Arithmetical truth is not arithmetically definable, i.e. the set of Gödel numbers of formulas that are true in natural number arithmetic are not definable in the theory of arithmetic.

If a theory $\mathcal{T}$ is decidable, then there is an effective method for deciding whether a given formula is a theorem in $\mathcal{T}$. One only has to calculate the Gödel number of that formula and test whether the value of the corresponding characteristic function is 1. Conversely, if a theory is undecidable, then there cannot be an effective method for checking whether a given formula is a theorem or not.

Theorem 1: No theory that can represent the computable functions is decidable.

Proof: Assume that $\mathcal{T}$ is a decidable theory that can represent the computable functions. Then the characteristic function of the set of Gödel numbers of $\mathcal{T}$ is representable, which means that the set of Gödel numbers of $\mathcal{T}$ is definable. Because of Lemma 2 this cannot be the case. □

Corollary: Arithmetic is undecidable.

An immediate consequence of Theorem 1 is Church's undecidability theorem, which states that first-order logic is undecidable. We can easily reduce the decidability problem for first-order logic to the decidability problem for theories that can represent the computable functions, provided we find a finitely axiomatizable theory that has this property. Since the theories we considered so far are not finitely axiomatizable, we postpone the proof of Church's undecidability theorem until we have introduced the theory $\mathcal{Q}$.

The following lemma provides the key argument for Gödel's First Incompleteness Result.

Lemma 3: Every axiomatizable and complete theory is decidable.

Proof: Let $\mathcal{T}$ be an axiomatizable complete theory and S be its set of axioms.
If $\mathcal{T}$ is inconsistent, then for some formula X both X and $\sim X$ are theorems of $\mathcal{T}$, thus consequences of the axioms. As a result, every formula in the language of $\mathcal{T}$ is a logical consequence of the axioms, which means that $\mathcal{T}$ is the set of all formulas, hence decidable.
Assume from now on that $\mathcal{T}$ is consistent, too. Since S is decidable, we can determine whether a given derivation of a formula from the axioms constitutes a valid logical inference. Therefore, it is possible to *enumerate* all the valid derivations in $\mathcal{T}$ and all the logical consequences of the axioms.
Now let X be an arbitrary formula in the language of $\mathcal{T}$. Since $\mathcal{T}$ is complete, the enumeration of the logical consequences of the axioms in S will yield either X or $\sim X$ after finitely many steps. Thus we can decide whether X is a theorem in $\mathcal{T}$ or not. $\square$

Theorem: [Gödel's First Incompleteness Theorem]

There is no consistent, complete, and axiomatizable theory that can represent the computable functions.

Proof: This is an immediate consequence of Lemma 3 and the fact that no theory that can represent the computable functions is decidable (Theorem 1). $\square$

Corollary: Arithmetic is not axiomatizable.

Gödel's incompleteness theorem is often described as "*any consistent and sufficiently strong formal theory of arithmetic is incomplete*", where a formal theory is viewed as one whose theorems are derivable from an axiom system. For such theories there will always be formulas that are true (, for instance, in the standard interpretation of arithmetic) but not theorems of the theories. When it comes to arithmetic, *truth and provability are in no sense the same.*

The above formulation is a sharpening of that statement, as it specifies "sufficiently strong" as "capable of representing computable functions". As we will see later, this includes theories that are much weaker than Peano Arithmetic.

Gödel's incompleteness result has a strong impact on the development of formal reasoning environments for mathematics. It is impossible to develop a consistent and complete proof system for any fragment of mathematics that includes arithmetic. As formal systems necessarily need to have decidable sets of axioms (even if we use axiom schemes), we have to sacrifice completeness, which means that there will always be theorems that are true but cannot be proven in the proof system.

On the other hand, one has to be aware that the incompleteness result has little effect on *practical theorem proving*. Most, if not all theorems in practical mathematics do not involve the issue of self-application and will therefore remain provable in a formal system.

Undecidability, however, is a bigger problem, as it shows the limitations of fully automated proof systems. While it is possible to find a proof for almost all (valid) practical problems in a formal proof system, the undecidability of first-order logic makes it impossible to check whether a proof attempt fails or just needs more time to complete. For this reason, all proof systems that aim at formalizing large fragments of mathematics and computer science (like Cornell's Nuprl system) proceed interactively with some controlled automated support for decidable sub-problems.

The undecidability of logic, the undefinability of arithmetical truth, and the incompleteness of systems of arithmetic are not the only unsolvable problems for expressive logics. A fourth issue involves the *undefinability of provability*: it is not possible to describe a predicate $Prov$ that represents provability in a theory $\mathcal{T}$ such that $\sim Prov(\lceil \texttt{false} \rceil)$ is a theorem in $\mathcal{T}$. We call a predicate $Prov$ a *provability predicate for $\mathcal{T}$* if $Prov$ satisfies the following conditions for all formulas X and Y.

- If $\models_T X$ then $\models_T Prov(\lceil X \rceil$
- $\models_T Prov(\lceil X \supset Y \rceil) \supset (Prov(\lceil X \rceil) \supset Prov(\lceil Y \rceil))$
- $\models_T Prov(\lceil X \rceil) \supset Prov(\lceil Prov(\lceil X \rceil) \rceil)$

The first condition states that every theorem should be provable, the second that the modus ponens holds for provability, and the third that provability is provable. Note that the second and third conditions are stronger than the first in the sense that the implication itself must be a theorem in $\mathcal{T}$.

Note that a condition like $\models_T Prov(\lceil X \rceil) \supset X$ is not included in the definition, since this requirement cannot be satisfied unless $\mathcal{T}$ is inconsistent. In fact, Löb's Theorem shows that this condition implies $\models_T X$.

Theorem: [Löb's Theorem]

If $Prov$ is a provability predicate for a theory $\mathcal{T}$ that can represent the computable functions then $\models_T Prov(\lceil X \rceil) \supset X$ implies $\models_T X$ for any X.

Proof: Assume $\models_T Prov(\lceil X \rceil) \supset X$. By the diagonal lemma, the formula $Prov(y) \supset X$ must have a fixed point G, i.e. there is a formula G such that $\models_T G \Leftrightarrow (Prov(\lceil G \rceil) \supset X)$. Since $Prov$ is a provability predicate, we know
$\models_T Prov(\lceil G \supset (Prov(\lceil G \rceil) \supset X) \rceil)$ and also
$\models_T Prov(\lceil G \rceil) \supset Prov(\lceil Prov(\lceil G \rceil) \supset X \rceil)$. and
$\models_T Prov(\lceil G \rceil) \supset Prov(\lceil Prov(\lceil G \rceil) \rceil) \supset Prov(\lceil X \rceil)$.

Since $\models_T Prov(\lceil G \rceil) \supset Prov(Prov(\lceil G \rceil))$ we get $\models_T Prov(\lceil G \rceil) \supset Prov(\lceil X \rceil)$.

Because of our assumption we may conclude $\models_T Prov(\lceil G \rceil) \supset X$ and $\models_T G$ follows from the definition of G. From this we may conclude $\models_T Prov(\lceil G \rceil)$, which finally gives us $\models_T X$. □

An immediate consequence of Löb's Theorem is Gödel's Second Incompleteness Theorem, which states that provability predicates for consistent predicates cannot be complete.

Theorem: [Gödel's Second Incompleteness Theorem]

If *Prov* is a provability predicate for a consistent theory $\mathcal{T}$ that can represent the computable functions then $\not\models_T \sim Prov(\lceil \underline{\mathbf{0}}\texttt{=}\underline{\mathbf{1}} \rceil)$.

Proof: Suppose $\models_T \sim Prov(\lceil \underline{\mathbf{0}}\texttt{=}\underline{\mathbf{1}} \rceil)$. By definition of negation $\models_T Prov(\lceil \underline{\mathbf{0}}\texttt{=}\underline{\mathbf{1}} \rceil) \supset \underline{\mathbf{0}}\texttt{=}\underline{\mathbf{1}}$. By Löb's theorem, we then conclude $\models_T \underline{\mathbf{0}}\texttt{=}\underline{\mathbf{1}}$, which cannot be because $\mathcal{T}$ is consistent. $\Box$