9 Incompleteness

In this chapter, we're going to prove several versions of Gödel's First Incompleteness Theorem. We're also going to prove Tarski's Theorem on the undefinability of truth, and Church's Theorem on the undecidability of first-order logic.

9.1 Preview

In Chapter 4, we studied axiomatic theories. The aim of axiomatizing an area of mathematics (or other discipline), is to put it on a firm foundation: instead of relying on a hodgepodge of intuition and imperfectly understood techniques, all results in the area should be derivable by pure logic from a set of precise and explicitly stated assumptions: the axioms.

What should we want of an axiomatic theory? Obviously, all the axioms should be true on their intended interpretation. Ideally, they should suffice to derive *all* truths in the relevant area. These are semantic properties of theories, related to their intended interpretation. But they entail syntactic properties. If all axioms of a theory are true, the theory must be *consistent*: it won't contain a sentence A and its negation $\neg A$; equivalently, it won't contain $\bot$. If a theory contains all truths on its intended interpretation, it will be *complete* in the sense that for every sentence A in its language, it contains either A or $\neg A$.

Note that this is not the sense of 'complete' in which the first-order calculus is complete. When we talk about completeness *of a proof system*, we mean that the system can prove every valid sentence. When we talk about completeness *of a theory*, we mean that the theory decides every sentence: it contains A or $\neg A$, for every sentence A. This notion of completeness is sometimes called *negation-completeness*.

Confusingly, 'sound' also has two meanings. A proof system is sound if it can only prove valid sentences. A theory is called *sound* if all sentences in the theory are true on their intended interpretation.

Consistency and completeness are defined syntactically, without reference to the meaning of the axioms. This makes them easier to study formally than semantic properties

like soundness, which requires pinning down the intended interpretation independently of the proposed axioms, so that one can compare what the axioms say with the structure they are meant to describe.

Consistency may seem trivial: surely nobody would propose an inconsistent set of axioms? But remember that this is exactly what happened to Frege. It also happened to others, especially when trying to develop powerful systems to unify diverse areas of mathematics. Many mathematicians were therefore wary of ZFC when it was first proposed. Couldn't it also turn out to be inconsistent?

David Hilbert saw how such fears could be put to rest. To check whether an axiomatic theory is consistent, we only need to check whether there is deduction of $\bot$ from its axioms. Even if the theory itself, like ZFC, talks about highly infinitary matters, any deduction from its axioms is finite. We should therefore be able to establish the consistency of ZFC in a much weaker, finitary branch of mathematics that doesn't study sets, but proofs and deductions. In the same way, Hilbert hoped that we could prove the completeness of axiomatic theories: we should be able to verify (as seemed plausible in the 1920s) that the axioms of, say, Peano Arithmetic or ZFC decide every sentence in their language.

This project for establishing the consistency and completeness of axiomatic theories is known as *Hilbert's program*. It was shattered by Gödel's incompleteness theorems. Gödel showed that sufficiently strong axiomatic theories can never be complete, unless they are inconsistent. He also showed that there is no hope of establishing the consistency of sufficiently strong theories from safe, finitary grounds. In the present chapter, we'll focus on the first of these results. We'll turn to the second in Chapter 10.

Gödel realized that we don't need a separate branch of mathematics to study proofs. Since sentences and deductions are finite strings of symbols, they can be coded as numbers. We can therefore use arithmetical theories to reason indirectly about proofs. We can, for example, construct an $\mathfrak{L}_A$-formula $\text{PROV}_{\text{PA}}(x)$ so that $\text{PROV}_{\text{PA}}(\overline{n})$ is true (in the standard model of arithmetic $\mathfrak{A}$) iff there is a deduction of the sentence coded by n from the axioms of Peano Arithmetic.

Gödel then showed how to construct a sentence G, coded by some number n, such that $G \leftrightarrow \neg\text{PROV}_{\text{PA}}(\overline{n})$ is true in $\mathfrak{A}$. Informally, G says of itself that it is not provable (in PA): it is true iff it is unprovable. It swiftly follows that PA can't decide G, assuming that PA is sound. To see this, note first that G is true: if it were is false, it would be provable (because G is true iff it is unprovable), and so PA would prove a falsehood, contradicting the assumption that PA is sound. So G is true. And so G can't be proved in PA (because G is true iff it is unprovable). Its negation $\neg G$ can't be proved either, as otherwise PA would prove a falsehood.

This argument assumes that PA is sound. For that reason, it is known as the "semantic" version of the First Incompleteness Theorem. Gödel's main result is a "syntactic" version of the theorem that doesn't require soundness. In its standard formulation, it shows that every consistent axiomatic theory that is at least as strong as Q is incomplete.

The restriction to *axiomatic* theories is crucial. Consider the theory $\mathrm{Th}(\mathfrak{A})$ consisting of all $\mathfrak{L}_A$-sentences that are true in the standard model of arithmetic $\mathfrak{A}$. This theory is complete: every $\mathfrak{L}_A$-sentence is either true or false in $\mathfrak{A}$; if true, it is in $\mathrm{Th}(\mathfrak{A})$; if false, its negation is in $\mathrm{Th}(\mathfrak{A})$. By definition, $\mathrm{Th}(\mathfrak{A})$ is also sound, and therefore consistent. But it is not an axiomatic theory: I haven't specified $\mathrm{Th}(\mathfrak{A})$ by giving a set of axioms, and the Incompleteness Theorem implies that I couldn't have done so.

Officially, theories are just sets of sentences closed under first-order consequence. The same set of sentences can always be specified in many ways. Instead of speaking of *axiomatic* theories, we should therefore speak of *axiomatizable* theories. Recall that a theory is axiomatizable if there is a decidable set of axioms from which all and only the sentences in the theory can be deduced: a set of axioms for which there is a mechanical algorithm to check whether a given sentence is in it or not.

In Section 5.4, I introduced these concepts informally, using the pre-theoretical notions of computability and algorithms. We can now give more formal definitions.

We say that a set of numbers is *recursively decidable* if the property of being in the set is a recursive relation, as defined in Chapter 7; a set of sentences is recursively decidable if the set of their code numbers is recursively decidable. A theory is *recursively axiomatizable* if there is a recursively decidable set of axioms from which all and only the sentences in the theory can be deduced. By Theorem 7.4, this is equivalent to saying that there is a Turing machine that determines whether a given $\mathfrak{L}_A$-sentence is among the axioms or not.

The syntactic version of Gödel's First Incompleteness Theorem can now be stated as follows: *every consistent and recursively axiomatizable theory of arithmetic that is at least as strong as Q is incomplete.*

Exercise 9.1 Let T_1 be the set of all $\mathfrak{L}_A$-sentences. Is T_1 (a) a theory? (b) recursively axiomatizable? (c) complete? (d) consistent? (Explain.)

Exercise 9.2 Let T_2 be the set of $\mathfrak{L}_A$-sentences that are valid in first-order logic. Is T_2 (a) a theory? (b) recursively axiomatizable? (c) complete? (d) consistent?

Exercise 9.3 Can you find an $\mathfrak{L}_A$-theory that is recursively axiomatizable, complete, and consistent? (Hint: you only need one simple axiom.)

9.2 Arithmetization of syntax

As I mentioned above, Gödel's proof draws on the insight that we can use arithmetical theories like PA to reason about their own syntax. After the work we've done in the previous chapter, this should not be surprising. We've shown in Theorem 8.5 that every computable property or relation is definable in $\mathfrak{L}_A$. Syntactic properties like *coding an $\mathfrak{L}_A$-sentence* or *coding a deduction from the axioms of PA* are clearly computable; so they are definable in $\mathfrak{L}_A$: there is an $\mathfrak{L}_A$-formula $\mathrm{PRF}_{\mathrm{PA}}(x, y)$ such that $\mathrm{PRF}_{\mathrm{PA}}(\overline{n}, \overline{m})$ is true (in $\mathfrak{A}$) iff n codes a proof of the sentence coded by m from the axioms of PA. This is all we need to run Gödel's argument. To fix ideas, I'll nonetheless fill in some more details.

We want to talk about sentences and deductions in the language $\mathfrak{L}_A$, whose non-logical symbols are 0, s, $+$, and $\times$. To this end, we code $\mathfrak{L}_A$-strings as numbers, so that we can indirectly refer an $\mathfrak{L}_A$-string by the $\mathfrak{L}_A$-numeral of its code. We'll use Gödel's own coding scheme, which I introduced in Section 5.5.

We first assign a *symbol code* to each primitive symbol of $\mathfrak{L}_A$, like so:

Symbol:	0	s	$+$	$\times$	$=$	$\neg$	$\rightarrow$	$\forall$	(	)	,	x_1	c_1	x_2	c_2	…
Code:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	…

Then we use the prime exponent method to code sequences of symbol codes, and thereby $\mathfrak{L}_A$-strings. The string '$0 = 0$', for example, determines the sequence of symbol codes $\langle 1, 5, 1 \rangle$, which is coded as $2^1 \cdot 3^5 \cdot 5^1 = 2430$. The exponents of the primes are the symbol codes. In general, if p_i is the ith prime number then the code number of an $\mathfrak{L}_A$-string A composed of symbols $s_1 s_2 \ldots s_n$ with symbol codes $c_1, c_2, \ldots, c_n$ is

$$\#[A] = p_1^{c_1} \cdot p_2^{c_2} \cdot \ldots \cdot p_n^{c_n}.$$

From now on, we'll call $\#[A]$ the *Gödel number* of A. Note that individual symbols of $\mathfrak{L}_A$ have both a Gödel number and a symbol code: '$\rightarrow$' has symbol code 7 and Gödel number $2^7 = 128$. We won't talk about symbol codes any more.

Since deductions are finite sequences of $\mathfrak{L}_A$-sentences, we can use the prime exponent method again to code them: the Gödel number of a deduction $A_1, A_2, \ldots, A_n$ is

$$\#[A_1, A_2, \ldots, A_n] = p_1^{\#[A_1]} \cdot p_2^{\#[A_2]} \cdot \ldots \cdot p_n^{\#[A_n]}.$$

The Gödel number function # converts any $\mathfrak{L}_A$-string A into a number $\#[A]$. This number $\#[A]$ is denoted in $\mathfrak{L}_A$ by some numeral $\overline{\#[A]}$. So we can indirectly refer to any $\mathfrak{L}_A$-string A by the numeral $\overline{\#[A]}$ of its Gödel number.

We'll abbreviate $\overline{\#[A]}$ as $\ulcorner A \urcorner$. For example, since $\#[0 = 0] = 2430$, $\ulcorner 0 = 0 \urcorner$ is $\overline{2430}$, which is $s(s(\ldots s(0) \ldots))$ with 2430 occurrences of s. In practice, you should treat the corner quotes as a special kind of quote marks: we use $\ulcorner 0 = 0 \urcorner$ to denote the string '0 = 0' via its Gödel number.

Exercise 9.4 What are (a) $\#[0]$? (b) $\overline{\#[0]}$? (c) $\ulcorner 0 \urcorner$? (d) $\#[\ulcorner 0 \urcorner]$?

Now consider a simple syntactic property: being a variable. In our coding scheme, variables have Gödel numbers $2^{12}, 2^{14}, 2^{16}, \ldots$. That is, a number n codes a variable iff $n = 2^{12+2y}$, for some y. This is a purely arithmetical property that can be expressed in $\mathfrak{L}_A$: there is an $\mathfrak{L}_A$-formula $\mathrm{VAR}(x)$ such that $\mathrm{VAR}(\overline{n})$ is true (in $\mathfrak{A}$) iff n codes a variable. In this sense, $\mathrm{VAR}(\overline{n})$ "says that" n codes a variable. But what it actually, explicitly says is simply that there is a number y such that $n = 2^{12+2y}$.

In the terminology of the previous chapter, the formula $\mathrm{VAR}(x)$ *defines* the property of coding a variable. By Theorem 8.5, every recursive relation and function is definable in L_A. We can use this result to show that a wide range of syntactic notions are definable in L_A. Since our coding scheme can be implemented mechanically, it maps every computable relation or function on $\mathfrak{L}_A$-strings to a computable relation or function on $\mathbb{N}$. By the Church-Turing Thesis, that relation or function is recursive. By Theorem 8.5, it is definable in L_A.

For example, there is a mechanical procedure for checking whether a given string is a well-formed sentence of $\mathfrak{L}_A$. So there is also a mechanical procedure for checking whether a given number is the Gödel number of an $\mathfrak{L}_A$-sentence. By the Church-Turing Thesis, it follows that the property (call it Sent) of coding an $\mathfrak{L}_A$-sentence is recursive. By Theorem 8.5, it follows that there is an $\mathfrak{L}_A$-formula $\mathrm{SENT}(x)$ such that $\mathrm{SENT}(\overline{n})$ is true (in $\mathfrak{A}$) iff n is the Gödel number of an $\mathfrak{L}_A$-sentence.

Similarly, if a theory T is axiomatized by a decidable set of axioms then there is a mechanical procedure for checking whether a given sequence of $\mathfrak{L}_A$-sentences is a deduction of a given target sentence from these axioms: we only need to check whether

the last sentence in the sequence is the target sentence, and whether each sentence in the sequence is either an axiom of T, an instance of the logical axioms A1–A6, or follows from previous sentences by MP or Gen. All these checks can be performed mechanically. Let Prf_T be the relation that holds between numbers n and m iff n codes a deduction (informally, a "proof") of the sentence coded by m from a set of axioms that generates T. If T is computably axiomatizable, Prf_T is computable. By the Church-Turing Thesis, it is recursive. By Theorem 8.5, it follows that it is definable in L_A: there is an $\mathfrak{L}_A$-formula $\text{PRF}_T(x, y)$ such that $\text{PRF}_T(\overline{n}, \overline{m})$ is true (in $\mathfrak{A}$) iff n codes a proof of the sentence coded by m from the axioms of T.

For a final example, let's look at a function on $\mathfrak{L}_A$-strings. Consider the *concatenation* function that takes two $\mathfrak{L}_A$-strings and returns the string consisting of the first followed by the second. This is clearly computable. By the Church-Turing Thesis, the corresponding function on Gödel numbers is recursive: there is a recursive function $*$ that maps the Gödel numbers of any two $\mathfrak{L}_A$-strings to the Gödel number of the concatenation of these strings. By Theorem 8.5, it follows that there is an $\mathfrak{L}_A$-formula $\text{CONCAT}(x, y, z)$ that defines $*$ in L_A, so that $\text{CONCAT}(\overline{n}, \overline{m}, \overline{k})$ is true (in $\mathfrak{A}$) iff k codes the concatenation of the strings coded by n and m. (I'll write this as $k = n * m$, rather than $k = *(n, m)$).

Exercise 9.5 What is $\#[0 =] * \#[0]$?

I've appealed to the Church-Turing Thesis to argue that Sent, Prf_T and $*$ are recursive, but we could have shown this directly: we could show that Sent, Prf_T and $*$ can be constructed from zero, successor, and projection by composition, primitive recursion, and regular minimization. In fact, we don't need minimization: Sent, Prf_T and $*$ are primitive recursive. I won't go through the details for each case. But let me illustrate what's involved with the concatenation function $*$ (which will play an important role in the next section).

Recall that $*$ maps two Gödel numbers $\#[A]$ and $\#[B]$ to the Gödel number $\#[AB]$ of the concatenation of A and B. If B is a single symbol, it is easy to define this operation arithmetically:

$$\#[A] * \#[s] = \#[A] \cdot \text{pri}(\text{len}(\#[A]))^{\#[s]},$$

where $\text{pri}(i)$ is the ith prime number and $\text{len}(n)$ is the length of the string coded by n. In Section 7.2, I showed that pri and len are primitive recursive. So the function

$$\text{append}(x, y) = x \cdot \text{pri}(\text{len}(x))^y$$

is also primitive recursive.

Next, we need the function entry that takes two numbers n and i and returns the exponent of the ith prime in the prime factorization of n. I showed in Section 7.2 that this function, too, is primitive recursive. Using append and entry, we define (by primitive recursion) a function conc that takes three numbers n, m, and i and returns the code of the string consisting of the string coded by n followed by the first i symbols of the string coded by m:

$$\begin{aligned} \mathrm{conc}(x, y, 0) &= x \\ \mathrm{conc}(x, y, s(i)) &= \mathrm{append}(\mathrm{conc}(x, y, i), \mathrm{entry}(y, s(i))). \end{aligned}$$

From this, we can define $x * y$ as $\mathrm{conc}(x, y, \mathrm{len}(y))$.

9.3 The First Incompleteness Theorem

I'll now explain how Gödel managed to construct a sentence that is true iff it is unprovable. The construction is so perplexing that it may help to first give a version for English. I'll show how to construct an English sentence that is true iff it is unprovable. (Let's pretend we've specified what it means for an English sentence to be "provable". You'll see that nothing hangs on this.)

In English, we can use quote marks to denote expressions of English. For example,

> 'is English'

is a noun that denotes an English predicate. We can combine nouns like this with predicates to form sentences:

(1) 'is English' is English.

(2) 'is made of stone' is made of stone.

(3) 'is made of stone' is English.

In (1) and (2), a predicate is applied to itself, using quote marks. Let's call a sentence that results by applying a predicate to itself in this manner the *diagonalization* of that predicate. So (1) is the diagonalization of 'is English'.

Now consider the predicate 'has a diagonalization that is not provable'. If we diagonalize *this* predicate, we get

(4) 'has a diagonalization that is not provable' has a diagonalization that is not provable.

This is a sentence. What does it say? Well, it says that the predicate it quotes has an unprovable diagonalization. Every predicate has a unique diagonalization. So (4) says that the diagonalization of the quoted predicate ('has a diagonalization that is not provable') is not provable. But (4) *is* the diagonalization of that predicate. So (4) says of itself that it is not provable.

This trick obviously generalizes. We can replace 'is not provable' by any predicate P. The argument shows that for any predicate English P, there is a sentence G that says of itself that it is P.

We'll now run this argument for $\mathfrak{L}_A$. We use open formulas $A(x)$ as predicates, and Gödel numerals $\ulcorner A(x) \urcorner$ (instead of quote marks) to refer to these predicates. For example, if $\mathrm{VAR}(x)$ is the formula that defines the property of coding a variable, then $\mathrm{VAR}(\ulcorner \mathrm{VAR}(x) \urcorner)$ is a sentence saying (falsely) that the code of $\mathrm{VAR}(x)$ codes a variable – equivalently: that $\mathrm{VAR}(x)$ is a variable. We might call $\mathrm{VAR}(\ulcorner \mathrm{VAR}(x) \urcorner)$ the diagonalization of $\mathrm{VAR}(x)$. However, it proves convenient to use a slightly more roundabout definition.

For any $\mathfrak{L}_A$-formula A, we define the *diagonalization* of A as the formula

$$\exists x(x = \ulcorner A \urcorner \wedge A).$$

If x is free in A, which is the only case we care about, this is logically equivalent to $A(\ulcorner A(x) \urcorner)$.

With this definition, constructing the diagonalization of a formula is a trivial mechanical task. Let diag be the corresponding function on Gödel numbers: diag takes the Gödel number of a formula as input and returns the Gödel number of the formula's diagonalization. This function is recursive. In fact, it is primitive recursive, and easily definable with the concatenation function $*$:

$$\mathrm{diag}(y) = \#[\exists x(x = \overline{y} \wedge] * y * \#[)].$$

By Theorem 8.5, all recursive functions are definable in $\mathfrak{L}_A$. So there is a formula $\mathrm{DIAG}(x, y)$ such that $\mathrm{DIAG}(\overline{n}, \overline{m})$ is true (in $\mathfrak{A}$) iff m codes the diagonalization of the formula coded by n. We use this formula to construct, for any formula $A(x)$ a sentence that "says of itself" that it has the property expressed by $A(x)$.

Lemma 9.1: The Semantic Diagonal Lemma

For every $\mathfrak{L}_A$-formula $A(x)$ there is a sentence G such that $\mathfrak{A} \Vdash G$ iff $\mathfrak{A} \Vdash A(\ulcorner G \urcorner)$.

Proof. Let $F(x)$ be the formula $\exists y(\text{DIAG}(x, y) \wedge A(y))$. Let G be the diagonalization of $F(x)$. So G is $\exists x(x = \ulcorner F(x) \urcorner \wedge F(x))$. This is logically equivalent to $F(\ulcorner F(x) \urcorner)$, which expands to $\exists y(\text{DIAG}(\ulcorner F(x) \urcorner, y) \wedge A(y))$. Since DIAG defines diag, G is true in $\mathfrak{A}$ iff there is a number n that codes the diagonalization of $F(x)$ and for which $A(\overline{n})$ is true (in $\mathfrak{A}$). The diagonalization of $F(x)$ is G. So G is true in $\mathfrak{A}$ iff $A(\overline{n})$ is true (in $\mathfrak{A}$) of the number n that codes G. In short G is true in $\mathfrak{A}$ iff $A(\ulcorner G \urcorner)$ is true in $\mathfrak{A}$. (If this proof baffles you, have another look at the English version above!) □

Now we're ready to prove the semantic version of Gödel's First Incompleteness Theorem. Let T be some recursively axiomatizable theory in $\mathfrak{L}_A$, so that there is a recursively decidable set of axioms Γ from which all and only the members of T can be deduced. I'll say that a sentence is *provable in T* if it is deducible from some such set Γ. As above, let Prf_T be the relation that holds between numbers n and m iff n codes a deduction of the sentence coded by m from Γ. As explained in the previous section, Prf_T is recursive; so is an $\mathfrak{L}_A$-formula $\text{PRF}_T(x, y)$ such that $\mathfrak{A} \Vdash \text{PRF}_T(\overline{n}, \overline{m})$ iff n codes a deduction from Γ of the sentence coded by m. Let $\text{PROV}_T(x)$ abbreviate $\exists y\, \text{PRF}_T(y, x)$. By construction, $\text{PROV}_T(\ulcorner A \urcorner)$ is true (in $\mathfrak{A}$) iff A is provable in T. So $\neg\text{PROV}_T(\ulcorner A \urcorner)$ is true iff A is unprovable in T. By diagonalising $\neg\text{PROV}_T(x)$, we get a sentence G that is true (in $\mathfrak{A}$) iff it is unprovable (in T).

Theorem 9.1: Gödel's First Incompleteness Theorem, semantic version

Every sound and recursively axiomatizable $\mathfrak{L}_A$-theory is incomplete.

Proof. Let T be a recursively axiomatizable $\mathfrak{L}_A$-theory. As I've just explained, there is then an $\mathfrak{L}_A$-formula $\text{PROV}_T(x)$ such that $\text{PROV}_T(\ulcorner A \urcorner)$ is true in $\mathfrak{A}$ iff A is provable in T. By the Semantic Diagonal Lemma (using $\neg\text{PROV}_T(x)$ for $A(x)$), there is a sentence G such that $\mathfrak{A} \Vdash G$ iff $\mathfrak{A} \Vdash \neg\text{PROV}_T(\ulcorner G \urcorner)$.

Suppose G is provable in T. Then $\mathfrak{A} \Vdash \text{PROV}_T(\ulcorner G \urcorner)$, and so $\mathfrak{A} \nVdash G$, contradicting our assumption that T is sound. So G is not provable in T. So $\mathfrak{A} \Vdash \neg\text{PROV}_T(\ulcorner G \urcorner)$, and so $\mathfrak{A} \Vdash G$. It follows that $\neg G$ isn't provable in T either, as otherwise T would prove a falsehood. □

This is a beautiful argument, although the conclusion isn't news to us: we've already derived it from the unsolvability of the Halting Problem in Section 5.5 (which, of course, wasn't known when Gödel published his result).

Exercise 9.6 Theorem 9.1 shows that there is a true sentence G that is not provable in a sound and recursively axiomatizable theory such as PA. Suppose we add G as a new axiom to PA. Is the resulting theory complete? Is it sound?

Exercise 9.7 Explain why there are infinitely many $\mathfrak{L}_A$-sentences that PA can't decide (assuming that PA is sound).

As I mentioned in Section 9.1, Gödel also proved a syntactic version of the Incompleteness Theorem that doesn't require the relevant theory to be sound (true in $\mathfrak{N}$), but merely imposes some syntactic conditions on it.

The idea is to run through the proof of Theorem 9.1 inside the theory T. Instead of relying on the equivalence of G and $\neg\text{PROV}_T(\ulcorner G \urcorner)$ in $\mathfrak{N}$, we'll use the fact that T can *prove* their equivalence: $\vdash_T G \leftrightarrow \neg\text{PROV}_T(\ulcorner G \urcorner)$. This requires a different version of the Diagonal Lemma, turning on the *representability* of diag in T, rather than on its definability. Recall that a (one-place) function f is representable in a theory T iff there is a formula $A(x, y)$ such that for all n,

(i) $\vdash_T A(\overline{n}, \overline{f(n)})$, and
(ii) $\vdash_T \forall y(A(\overline{n}, y) \rightarrow y = \overline{f(n)})$.

Equivalently: $\vdash_T \forall y(A(\overline{n}, y) \leftrightarrow y = \overline{f(n)})$.

Lemma 9.2: The Syntactic Diagonal Lemma

If T is an $\mathfrak{L}_A$-theory in which diag is representable, then for every $\mathfrak{L}_A$-formula $A(x)$ there is a sentence G such that $\vdash_T G \leftrightarrow A(\ulcorner G \urcorner)$.

Proof. Let T be an $\mathfrak{L}_A$-theory in which diag is representable. Let $\text{DIAG}(x, y)$ be the formula that represents diag in T, and let $F(x)$ be the formula $\exists y(\text{DIAG}(x, y) \wedge A(y))$. Since DIAG represents diag in T, T can prove

$$\forall y\, (\text{DIAG}(\ulcorner F(x) \urcorner, y) \;\leftrightarrow\; y = \overline{\text{diag}(\ulcorner F(x) \urcorner)}). \tag{1}$$

Let G be the diagonalization of $F(x)$. So the following is logically true:

$$G \;\leftrightarrow\; \exists y(\text{DIAG}(\ulcorner F(x) \urcorner, y) \wedge A(y)). \tag{2}$$

From (1) and (2), first-order logic yields

$$G \leftrightarrow \exists y\,(y = \overline{\mathrm{diag}(\ulcorner F(x) \urcorner)} \wedge A(y)).$$

Since $\overline{\mathrm{diag}(\ulcorner F(x) \urcorner)}$ is $\ulcorner G \urcorner$, this simplifies to $G \leftrightarrow \exists y\,(y = \ulcorner G \urcorner \wedge A(y))$ and further to $G \leftrightarrow A(\ulcorner G \urcorner)$. □

Now assume that T is a recursively axiomatizable theory in which both diag and Prf_T are representable. As before, define $\mathrm{PROV}_T(x)$ as $\exists y\,\mathrm{PRF}_T(y, x)$. The Syntactic Diagonal Lemma gives us a sentence G (called the *Gödel sentence* for T) such that

$$\vdash_T G \leftrightarrow \neg\mathrm{PROV}_T(\ulcorner G \urcorner). \tag{D}$$

Let's go through the reasoning in the proof of Theorem 9.1 to show that T can't decide G.

One of the two directions goes through smoothly: we can show that G isn't provable in T, unless T is inconsistent. For suppose T can prove T. This means that there is a deduction of G from a suitable set of axioms for T. Since $\mathrm{PRF}_T(x, y)$ represents Prf_T in T, it follows that there is a number n (the code of the deduction) such that $\vdash_T \mathrm{PRF}_T(\overline{n}, \ulcorner G \urcorner)$. Since T is closed under first-order consequence, it follows that $\vdash_T \mathrm{PROV}_T(\ulcorner G \urcorner)$. By (D), we have $\vdash_T \neg G$, So T proves both G and $\neg G$.

The other direction is trickier. Suppose T can prove $\neg G$. By (D), T can then prove $\mathrm{PROV}_T(\ulcorner G \urcorner)$, which is short for $\exists y\,\mathrm{PRF}_T(y, \ulcorner G \urcorner)$. If T is consistent, there is no deduction of G from T's axioms. So $\mathrm{Prf}_T(n, \ulcorner G \urcorner)$ is false for every number n. Since $\mathrm{PRF}_T(x, y)$ represents Prf_T in T, it follows that $\vdash_T \neg\mathrm{PRF}_T(\overline{n}, \ulcorner G \urcorner)$ for every number n.

We now have the following situation: T proves $\exists y\,\mathrm{PRF}_T(y, \ulcorner G \urcorner)$, but also $\neg\mathrm{PRF}_T(\overline{n}, \ulcorner G \urcorner)$ for every number n. The theory says that *there is* a number of a certain kind, but also denies that any particular number 0,1,2,…is of that kind. This isn't inconsistency, but it is almost as bad. Gödel called it "ω-inconsistency": a theory is *ω-inconsistent* if there is a formula $A(x)$ such that

(i) $\vdash_T \exists x\,A(x)$, but

(i) for every number n, $\vdash_T \neg A(\overline{n})$.

A theory is *ω-consistent* if it is not ω-inconsistent.

Clearly, no sound theory can be ω-inconsistent. So ω-consistency is another purely syntactic condition (besides consistency) that is entailed by soundness.

We've established the main result of Gödel's 1931 paper:

Theorem 9.2: Gödel's First Incompleteness Theorem

Every recursively axiomatizable and ω-consistent theory in which all recursive functions are representable is incomplete.

I won't go through the details of the proof again, as we're going to prove a strictly stronger result in the next section, showing that mere consistency (as opposed to ω-consistency) is enough. We will derive this from another important result, Tarski's Theorem. But I want to mention that there is also a way to establish it directly, following Gödel's line of reasoning. The trick, due to J. Barkley Rosser, is to make a slight change to the sentence G. Instead of using a sentence that says of itself that it is unprovable, Rosser uses a sentence saying that for every proof of it, there is a shorter proof of its negation. More formally, Rosser's version of the argument uses the diagonalization R of the following formula in place of G:

$$\forall y(\mathrm{PRF}_T(y,x) \rightarrow \exists z(z<y \wedge \forall v(\mathrm{CONCAT}(\ulcorner\neg\urcorner, x, v) \rightarrow \mathrm{PRF}_T(z,v))).$$

One can show that if Prf_T and diag are representable in T, T is consistent, and T knows a few facts about arithmetic, then it can prove neither R nor $\neg R$.

Exercise 9.8 Let G be the Gödel sentence for PA. We know that G is not provable in PA. How about $\mathrm{PROV}_{\mathrm{PA}}(\ulcorner G\urcorner)$? How about $\neg\mathrm{PROV}_{\mathrm{PA}}(\ulcorner G\urcorner)$?

Exercise 9.9 Explain why $\mathrm{PROV}_{\mathrm{PA}}(x)$ doesn't represent provability in PA. (Hint: use the previous exercise.)

Exercise 9.10 Show that every ω-consistent theory is consistent.

Exercise 9.11 Let T be an ω-inconsistent, but consistent theory. By the completeness of first-order logic, T has a model. Can you describe what such a model might look like?

9.4 Tarski's Theorem

Recall that a formula $A(x)$ represents a property P in a theory T iff for every $\mathfrak{L}_A$-sentence B,

(i) if $P(B)$, then $\vdash_T A(\ulcorner B \urcorner)$, and
(ii) if $\neg P(B)$, then $\vdash_T \neg A(\ulcorner B \urcorner)$.

In exercise 9.9, you showed that $\text{PROV}_{\text{PA}}(x)$ does not represent provability in PA. Officially, PA is just the set of all sentences that are provable in PA. You therefore showed that $\text{PROV}_{\text{PA}}(x)$ does not represent membership in PA.

This result can be strengthened. The following theorem, due to Alfred Tarski (1933), shows that no $\mathfrak{L}_A$-formula represents membership in PA. Indeed, no formula represents membership in any consistent theory in which diag is representable.

Theorem 9.3: Tarski's Theorem

If T is consistent and diag is representable in T, then membership in T is not representable in T.

Proof. Suppose $T(x)$ represents membership in T. By the Diagonal Lemma, there is a sentence G such that

$$\vdash_T G \leftrightarrow \neg T(\ulcorner G \urcorner) \tag{1}$$

Since $T(x)$ represents membership in T, we have

$$\text{if } \vdash_T G, \text{ then } \vdash_T T(\ulcorner G \urcorner) \tag{2}$$
$$\text{if } \nvdash_T G, \text{ then } \vdash_T \neg T(\ulcorner G \urcorner) \tag{3}$$

Either $\vdash_T G$ or $\nvdash_T G$. Suppose $\vdash_T G$. Then $\vdash_T \neg T(\ulcorner G \urcorner)$ by (1), and $\vdash_T T(\ulcorner G \urcorner)$ by (2); so T is inconsistent. Alternatively, suppose $\nvdash_T G$. Then $\vdash_T T(\ulcorner G \urcorner)$ by (1), and $\vdash_T \neg T(\ulcorner G \urcorner)$ by (3); again, T is inconsistent. □

Note that Tarski's Theorem isn't restricted to axiomatizable theories. It even holds for $\text{Th}(\mathfrak{N})$. Since representability in $\text{Th}(\mathfrak{N})$ implies definability in $\mathfrak{L}_A$, it follows that no $\mathfrak{L}_A$-formula defines membership in $\text{Th}(\mathfrak{N})$:

Theorem 9.4

Arithmetical truth is not definable in $\mathfrak{L}_A$: there is no $\mathfrak{L}_A$-formula $T(x)$ such that $\mathfrak{A} \Vdash T(\ulcorner A \urcorner)$ iff $\mathfrak{A} \Vdash A$.

Proof. $\mathrm{Th}(\mathfrak{A})$ is consistent extension of Q. By Theorem 8.3, it follows that diag is representable in $\mathrm{Th}(\mathfrak{A})$. By Theorem 9.3, it follows that membership in $\mathrm{Th}(\mathfrak{A})$ is not representable in $\mathrm{Th}(\mathfrak{A})$: there is no $\mathfrak{L}_A$-formula $T(x)$ such that

(i) if $\mathfrak{A} \Vdash A$ then $\mathfrak{A} \Vdash T(\ulcorner A \urcorner)$, and
(ii) if $\mathfrak{A} \nVdash A$ then $\mathfrak{A} \Vdash \neg T(\ulcorner A \urcorner)$.

So there is no $\mathfrak{L}_A$-formula that defines truth in $\mathfrak{A}$. □

Exercise 9.12 Use the Semantic Diagonal Lemma to prove Theorem 9.4, without invoking Theorem 9.3.

Tarski's Theorem shows that while $\mathfrak{L}_A$ can formalize its own syntax (we can define $\mathfrak{L}_A$-properties like being a variable or being a sentence), it can't express the most basic concept of its own semantics. This isn't just true for $\mathfrak{L}_A$. Loosely speaking, no sufficiently powerful language that can express its own syntax can express its own semantics.

We can bring this out a little more clearly by considering the concept of a truth predicate. As Tarski pointed out, the central feature of the predicate 'is true' in English is that when it is applied to a sentence, the result is equivalent to that sentence:

(1) 'Snow is white' is true iff snow is white.
(2) '2+2=4' is true iff 2+2=4.

Sentences like (1) and (2) are called *Tarski biconditionals*. A theory that can reason about truth should be able to prove all Tarski biconditionals for its language. Thus a formula $W(x)$ is called a *truth predicate for a theory* T iff $\vdash_T W(\ulcorner A \urcorner) \leftrightarrow A$ for every sentence A in T's language. An argument similar to the one used in Theorem 9.3 shows that no sufficiently powerful theory can have a truth predicate, unless it is inconsistent. This result is also called "Tarski's Theorem".

Theorem 9.5: Also Tarski's Theorem

If diag is representable in a consistent theory T then T has no truth predicate.

Proof. Suppose $W(x)$ is a truth predicate for T. By the Syntactic Diagonal Lemma, there is a sentence L such that $\vdash_T L \leftrightarrow \neg W(\ulcorner L \urcorner)$. Since $W(x)$ is a truth predicate for T, $\vdash_T W(\ulcorner L \urcorner) \leftrightarrow L$. So $\vdash_T L \leftrightarrow \neg L$. So T is inconsistent. □

While Gödel's sentence G says of itself that it is unprovable, the sentence L that figures in this proof says of itself that it is not true. It is a formal analogue of the Liar sentence 'This sentence is false'. The existence of such a sentence leads to paradox: if L is true then L is false, and if L is false then L is true. Theorem 9.5 concludes that L can't exist. By the Diagonal Lemma, it would exist if there were a truth predicate for T. So there can be no truth predicate for T. By contrast, it is not an option to deny the existence of G. By the Diagonal Lemma, G can be constructed from $\text{PROV}_T(x)$. The existence of $\text{PROV}_T(x)$ is guaranteed by the fact that (for suitable choices of T) Prf_T is recursive.

Exercise 9.13 Show that if T is a sound theory then there is no truth predicate for T.

We'll now use Tarski's Theorem to derive both the undecidability of first-order logic and strengthened versions of Gödel's First Incompleteness Theorem. We begin with two small lemmas.

Lemma 9.3

Every consistent theory in which all recursive functions are representable is recursively undecidable.

Proof. Let T be a consistent theory in which all recursive functions are representable. By Theorem 9.3, membership in T is not representable in T. So membership in T is not recursive: the set of Gödel numbers of sentences in T is not recursively decidable. □

Lemma 9.4

Let $\hat{Q}$ be the conjunction of the seven axioms of Q. The set of $\mathfrak{L}_A$-sentences of the form $\hat{Q} \rightarrow A$ is recursively decidable.

Proof. Let P be the property of coding sentences of the form $\hat{Q} \rightarrow A$. $P(n)$ can be defined as $\exists y \leq n\,(\text{Sent}(y) \wedge (n = \ulcorner \hat{Q} \rightarrow \urcorner * y))$. So $P(n)$ is (primitive) recursive. □

Theorem 9.6: Church's Theorem
The set of valid first-order sentence is recursively undecidable.

Proof. By Theorem 8.3, all recursive functions are representable in Q. Since Q is consistent, it follows by Lemma 9.3 that Q is recursively undecidable. As in the previous proof, let $\hat{Q}$ be the conjunction of Q's axioms. If the set of valid first-order sentences were recursively decidable, the set of valid $\mathfrak{L}_A$-sentences of the form $\hat{Q} \to A$ would also be recursively decidable, by Lemma 9.4 and the fact that the intersection of two recursively decidable sets is recursively decidable. By the soundness and completeness of first-order logic, $\hat{Q} \to A$ is valid iff $\vdash_Q A$. So Q would be recursively decidable, contradicting what we just established. □

Church's Theorem shows that Hilbert's Entscheidungsproblem has no solution: there is no mechanical procedure that decides whether an arbitrary first-order sentence is valid.

Exercise 9.14 In Section 6.4, I explained how Theorem 9.6 can be derived from the unsolvability of the Halting Problem. Explain in outline how we could derive the unsolvability of the Halting Problem from Theorem 9.6. (Hint: Given a first-order sentence A, we could mechanically go through all first-order proofs until we find a proof of A, in which case we halt and output 'yes'.)

Exercise 9.15 Explain why there can be no recursive bound on the length of a proof for a sentence in the first-order calculus: for every recursive function f, there is a sentence with length n that is provable, but whose proof requires more than $f(n)$ lines.

Now for the strengthened version of the syntactic Incompleteness Theorem. In Section 5.4 (Proposition 5.5), I showed that every axiomatizable and complete first-order theory is decidable. Together with Lemma 9.3, this immediately gives us the what we seek: every axiomatizable and consistent theory in which all recursive functions are representable is incomplete.

However, Proposition 5.5 used the informal concept of computable axiomatizability; so this argument relies on the Church-Turing Thesis. Let's prove a parallel result for recursive axiomatizability.

Lemma 9.5

Every recursively axiomatizable and complete first-order theory is recursively decidable.

Proof. Let T be a recursively axiomatizable and complete first-order theory. As in the previous section, let Prf_T be the relation that holds between numbers n and m iff n codes a deduction of m from some recursively decidable set of axioms for T. We know that Prf_T is recursive. We can now define the property W of coding a member of T as follows:

$$W(x) \text{ iff } \mathrm{Prf}_T(\mu p[\mathrm{Prf}_T(p,x) \vee \mathrm{Prf}_T(p,\#[\neg] * x)],x)$$

To see how this works, let A be any sentence, and x its Gödel number. $\mu p[\mathrm{Prf}_T(p,x) \vee \mathrm{Prf}_T(p,\#[\neg] * x)]$ finds the (Gödel number of the) first proof of either A or $\neg A$ in T. Since at least one of A and $\neg A$ must be in T by completeness, this search is guaranteed to terminate. The outer Prf_T then checks whether the proof that has been found is a proof of A. □

Theorem 9.7: Also Gödel's First Incompleteness Theorem

Every consistent and recursively axiomatizable theory in which all recursive functions are representable is incomplete.

Proof. Let T be a consistent and recursively axiomatizable theory in which all recursive functions are representable. By Lemma 9.3, T is undecidable. By Lemma 9.5, it follows that T is incomplete. □

Exercise 9.16 Let T be a consistent theory in which diag is representable. By Theorem 9.3, there is no formula $W(x)$ such that

(i) if $\vdash_T A$, then $\vdash_T W(\ulcorner A \urcorner)$, and
(ii) if $\nvdash_T A$, then $\vdash_T \neg W(\ulcorner A \urcorner)$.

But there could be formula $W(x)$ such that

(i*) $\vdash_T A$ iff $\vdash_T W(\ulcorner A \urcorner)$.

In the terminology of exercise 8.16, this formula *weakly represents* membership in T. Show that if such a formula exists then T is incomplete.

(Hint: use the Diagonal Lemma to infer that there is a sentence G such that $\vdash_T G \leftrightarrow \neg W(\ulcorner G \urcorner)$; show that neither G nor $\neg G$ is in T.)

Since all recursive functions are representable in every extension of Q (Theorem 8.3), Theorem 9.7 is often stated as saying that every consistent and recursively axiomatizable extension of Q is incomplete. We can prove an even stronger result by strengthening Lemma 9.3.

Lemma 9.6

Every $\mathfrak{L}_A$-theory consistent with Q is recursively undecidable.

Proof. Let T be an $\mathfrak{L}_A$-theory consistent with Q, and suppose for reductio that T is recursively decidable: the set W of Gödel numbers of sentences in T is recursive. Since diag is recursive, so is the property P that holds of a number x iff $\mathrm{diag}(x)$ is not in W. By exercise 8.16.(c), all recursive relations are weakly representable in any $\mathfrak{L}_A$-theory consistent with Q. So P is weakly representable in T: there is a formula $A(x)$ such that $\vdash_T A(\overline{n})$ iff $P(n)$. So:

$$\begin{aligned} \vdash_T A(\ulcorner A(x) \urcorner) &\text{ iff } P(\#[A(x)]) \\ &\text{iff } \mathrm{diag}(\#[A(x)]) \notin W \\ &\text{iff } \#[\exists x(x = \ulcorner A(x) \urcorner \wedge A(x))] \notin W \\ &\text{iff } \nvdash_T \exists x(x = \ulcorner A(x) \urcorner \wedge A(x)) \\ &\text{iff } \nvdash_T A(\ulcorner A(x) \urcorner). \end{aligned}$$

Contradiction. □

Theorem 9.8

Every recursively axiomatizable $\mathfrak{L}_A$-theory that is consistent with Q is incomplete.

Proof. Let T be a recursively axiomatizable $\mathfrak{L}_A$-theory that is consistent with Q. By Lemma 9.6, T is recursively undecidable. By Lemma 9.5, it follows that T is incomplete. □

9.5 The arithmetical hierarchy

Let's take stock. Since all recursive functions are representable in PA, Gödel's Theorem shows that PA is incomplete (unless it is inconsistent). The incompleteness can't be fixed by simply adding more axioms: as long as the resulting theory is consistent and axiomatizable, it will remain incomplete.

The result carries over to more powerful theories like ZFC, in virtue of the fact that PA is interpretable in these theories (see Section 4.3). More generally, Gödel's Theorem applies whenever a theory's language is rich enough to express central aspects of its own syntax. This isn't always the case. For example, consider a fragment of $\mathfrak{L}_A$ whose only non-logical symbols are 0, s, and $+$, without $\times$. In the previous chapter, we needed multiplication to define the recursive functions and relations. Without multiplication, Prf_T and $*$ are no longer definable. As a consequence, the Incompleteness Theorems don't apply. Indeed, if you restrict the axioms of PA to this weaker language, and remove the two axioms for multiplication, you get a complete theory. (This theory is called *Presburger Arithmetic*.)

Return to PA. We know that there are (infinitely many) true sentences that PA can't prove. But what do they look like? This is important to assess the practical significance of Gödel's result. If PA can't prove that 2+2=4, that's a serious problem. If the only arithmetical truths that PA can't prove take a trillion years to state, incompleteness may be harmless in practice.

Gödel's original proof (unlike the proof via Tarski's Theorem) gives us an example of an unprovable sentence: the "Gödel sentence" G. As I'll explain below, this sentence states (in a very roundabout way) that a certain complicated equation between polynomials has no solution in the natural numbers. If it weren't for Gödel's Theorem, no one would ever have considered this equation. Until the 1960s, the only sentences known to be unprovable in PA were of this kind. Since then, more natural examples have been found. The simplest is probably Goodstein's Theorem. (See Section 7.3.) Goodstein's Theorem states an interesting fact about the natural numbers, but its proof involves transfinite ordinals: it is provable in ZFC, but not in PA. For ZFC itself, we already know of a "natural" statement that it can't decide: the Continuum Hypothesis. There are many other examples.

To get a sense of which $\mathfrak{L}_A$-sentences are provable and which might be unprovable in PA, it is useful to classify the $\mathfrak{L}_A$-sentences by their construction from atomic formulas. Since the only predicate letter in $\mathfrak{L}_A$ is the identity predicate '=', all atomic formulas of $\mathfrak{L}_A$ are identity statements: they have the form $t_1 = t_2$. From these, complex formulas are constructed using truth-functional connectives and quantifiers. We'll divide them

into stages.

At the first stage, we have all identity statements $t_1 = t_2$, all inequalities of the form $t_1 < t_2$, and all formulas that can be constructed from these by truth-functional connectives and bounded quantification, where a bounded quantification of a formula A is a formula of the form $\forall x(x < t \rightarrow A)$ or $\exists x(x < t \wedge A)$, with x not occurring in t. (Officially, of course, $t_1 < t_2$ is short for $\exists z(t_1 + s(z) = t_2)$.) The formulas in this class are called Δ_0-*formulas*. Intuitively, a Δ_0-formula is any $\mathfrak{L}_A$-formula that doesn't involve unbounded quantification.

At the next stage, we consider all sentences that can be formed from Δ_0-formulas by prefixing unbounded universal quantifiers or unbounded existential quantifiers. A Δ_0-formula with a string of universal quantifiers in front is called a Π_1-*formula*; a Δ_0-formula with a string of existential quantifiers in front is called a Σ_1-*formula*. For example, $\forall x \forall y(x + y = y + x)$ is a Π_1-formula, while $\exists x \exists y(x + y = y + x)$ is a Σ_1-formula.

Prefixing universal quantifiers to a Σ_1-formula yields a Π_2-formula; prefixing existential quantifiers to a Π_1-formula yields a Σ_2-formula. Thus $\forall x \exists y(x + y = y + x)$ is Π_2, and $\exists x \forall y(x + y = y + x)$ is Σ_2. And so on.

This somewhat complicated classification is motivated by the computational properties of the relations defined by the relevant formulas. The relations expressed by Δ_0-formulas are all primitive recursive. Since Δ_0-formulas don't involve unbounded quantification, one can check whether they hold of some numbers by simple checks, without unbounded loops. By contrast, to check whether a Σ_1-formula $\exists x A(x)$ holds of some number, one may need to search through all numbers until one finds a witness for $A(x)$. Many Σ_1-formulas therefore express relations that are not primitive recursive. Some of them are merely recursive. In fact, every recursive relation is definable by a Σ_1-formula. But not every relation defined by a Σ_1-formula is recursive.

Some are just recursively enumerable. A relation R is *recursively enumerable* if there is a recursive relation S such that $R(x_1, \ldots, x_n)$ holds iff $\exists y\, S(x_1, \ldots, x_n, y)$. By the Church-Turing Thesis, the recursively enumerable relations are precisely the computably enumerable relations. (See Propositions 5.2 and 5.3 in Section 5.4.)

Theorem 9.9

A relation is recursively enumerable iff is definable in $\mathfrak{L}_A$ by a Σ_1-formula.

Proof sketch. I assume for readability that R is one-place.

From right to left, assume that R is defined by a Σ_1-formula $\exists y A(x, y)$, where A is Δ_0.

We can then mechanically enumerate all n for which $R(n)$ holds by going through all pairs of numbers (n, m) and check whether $A(n, m)$ holds.

For the other direction, we need show that every recursive relation is defined by a Σ_1-formula. Since prefixing existential quantifiers to a Σ_1-formula yields another Σ_1-formula, the result extends to every recursively enumerable relation.

The proof that every recursive function is defined by a Σ_1-formula proceeds by induction on the construction of recursive functions. In chapter 8, I showed that the base functions (zero, successor, projection) are definable in $\mathfrak{L}_A$, and that definability-in-$\mathfrak{L}_A$ is closed under composition, primitive recursion, and minimization. By going through each part of this proof, we can check that the defining formulas are all Σ_1. This is obvious for the base functions, which I explicitly defined using Δ_0-formulas. (For example, I showed that zero is defined by $x = 0$.) Closure under composition is also straightforward. I showed that $\mathrm{Cn}[f, g_1]$ is defined by $\exists v_1 (F(y, v_1) \wedge G_1(v_1, x_1, \ldots, x_n))$. Since any initial existential quantifiers in F and G_1 can simply be pulled to the front, so the whole formula is Σ_1 if F and G_1 are.

Regular minimization requires a more work. I showed that $\mathrm{Mn}[f]$ is defined by $F(x, y, 0) \wedge \forall z(z < y \rightarrow \neg F(x, z, 0))$. We need to show that any initial existential quantifiers in F can be pulled to the front. This is possible because $\forall z(z < y \rightarrow \neg \exists w F(x, z, 0))$ is equivalent to $\exists c \forall z(z < y \rightarrow \neg F(x, z, \mathrm{BETA}(c, z))$: the beta term $\mathrm{BETA}(c, z)$ retrieves the witness for $F(x, z, 0)$ from the code c. By going through the construction of BETA, one can show that it is definable by a Δ_0-formula.

Finally, for primitive recursion, I showed that $\mathrm{Pr}[f, g_1]$ is defined by $\exists c(\mathrm{SEQ}(c, x, k) \wedge \mathrm{BETA}(c, s(k), y))$, where $\mathrm{SEQ}(c, x, k)$ is defined in terms of F and G_1 and BETA. I've already mentioned that BETA is definable by a Δ_0-formula. Using the beta function trick that we've just used for minimization, one can show that $\mathrm{SEQ}(c, x, k)$ is definable by a Σ_1-formula, by pulling existential quantifiers to the front. □

We can use Theorem 9.9 to get an idea of what the unprovable Gödel sentence G for PA might look like. Recall that G is equivalent in PA to $\neg \mathrm{Prov}_{\mathrm{PA}}(\ulcorner G \urcorner)$. Since $\mathrm{Prov}_{\mathrm{PA}}$ is defined by existential quantification from the recursive relation $\mathrm{Prf}_{\mathrm{PA}}$, it is recursively enumerable. By Theorem 9.9, it is definable by a Σ_1-formula. So the Gödel sentence G is equivalent in PA to the negation of a Σ_1-sentence. This makes it equivalent to a Π_1-sentence. Gödel's result therefore shows that there are undecidable Π_1-sentences.

Theorem 9.9 can be strengthened:

Theorem 9.10: The MRDP Theorem

A relation is recursively enumerable iff it is definable in $\mathfrak{L}_A$ by a formula of the form $\exists x_1 \ldots \exists x_n\ t_1 = t_2$.

Since every $\mathfrak{L}_A$-term expresses a polynomial, the MRDP Theorem shows that every recursively enumerable relation is expressed by a formula stating that a certain equation between polynomials has a solution in the natural numbers. That's why I said that the Gödel sentence is equivalent to the statement that some equation between polynomials has no solution in the natural numbers. The proof of the MRDP theorem is too difficult to be even sketched here.

Let's return once more to Tarski and Gödel. By Theorem 9.4, arithmetical truth is not definable in $\mathfrak{L}_A$. With our new understanding of the arithmetical hierarchy, we can now strengthen the semantic Incompleteness Theorem. As stated in Theorem 9.1, the semantic Theorem says that every sound and recursively axiomatizable $\mathfrak{L}_A$-theory is incomplete. It is easy to see that a theory is recursively axiomatizable iff the set of Gödel numbers of its members is recursively enumerable. Theorem 9.1 therefore applies to all theories whose members are defined by a Σ_1-formula. We can extend the result to non-axiomatizable theories that are only definable by Π_2-formulas or Σ_{12}-formulas.

Let's say that a theory T is *definable in* $\mathfrak{L}_A$ if there is an $\mathfrak{L}_A$-formula $W(x)$ such that for all sentences B, $\mathfrak{A} \Vdash W(\ulcorner B \urcorner)$ iff $B \in T$.

Theorem 9.11

Every sound $\mathfrak{L}_A$-theory that is definable in $\mathfrak{L}_A$ is incomplete.

Proof. If T is sound and complete then $T = \mathrm{Th}(\mathfrak{A})$. By Theorem 9.4, $\mathrm{Th}(\mathfrak{A})$ is not definable in $\mathfrak{L}_A$. So if T is sound and definable in $\mathfrak{L}_A$ then it is incomplete. □

Exercise 9.17 Explain why a theory is recursively axiomatizable iff the set of Gödel numbers of its members is recursively enumerable. (Hint: if T is recursively axiomatizable then Prf_T is recursive.)