An Introduction to

Gödel's Theorems

Peter Smith

Faculty of Philosophy
University of Cambridge

Version date: December 12, 2005

The book's website is at www.godelbook.net

Contents

Preface

In 1931, the young Kurt Gödel published his First and Second Incompleteness Theorems; very often, these are simply referred to as 'Gödel's Theorems'. His startling results settled (or at least, seemed to settle) some of the crucial questions of the day concerning the foundations of mathematics. They remain of the greatest significance for the philosophy of mathematics – though just what that significance is continues to be debated. It has also frequently been claimed that Gödel's Theorems have a much wider impact on very general issues about language, truth and the mind. This book gives outline proofs of the Theorems and related formal results, and touches on some of their implications.

Who is this book for? Roughly speaking, for those who want a lot more detail than you get in books for a general audience (the best of those is Franzén 2005), but who find classic texts in mathematical logic (like Mendelson 1997) daunting and too short on explanatory scene-setting. So I hope philosophy students taking an advanced logic course will find the book useful, as will mathematicians who want a relatively relaxed exposition.

I originally intended to write a rather shorter book, leaving more of the formal details to be filled in from elsewhere. But while that plan might have suited some readers, I soon realized that it would seriously irritate others to be sent hither and thither to consult a variety of text books with different terminologies and different notations. So in the end, I have given more or less full proofs of most key results. However, my original plan shows through in two ways. First, some proofs are still only roughly sketched in, and there are other proofs which are omitted entirely. Second, I try to signal very clearly when the detailed proofs I do give can be skipped without much loss of understanding. With judicious skimming, you should be able to follow the main formal themes of the book even if you start from a very modest background in logic.[1]

As we go through, there is also an amount of broadly philosophical commentary. I follow Gödel in believing that our formal investigations and our general reflections on foundational matters should illuminate and guide each other. I hope that the more philosophical discussions (though certainly not always uncontentious) will also be reasonably widely accessible.

Writing a book like this presents many problems of organization. At various points we will need to call upon some background ideas from general logical theory. Do we explain them all at once, up front? Or do we introduce them as

[1] I plan, in due course, to put optional exercises (and answers!) on the book's website at www.godelbook.net.

we go along, when needed? Similarly we will also need to call upon some ideas from the general theory of computation – for example, we will make use of both the notion of a 'primitive recursive function' and the more general notion of a 'recursive function'. Again, do we explain these together? Or do we give the explanations many chapters apart, when the respective notions first get used?

I've mostly adopted the second policy, introducing new ideas as and when needed. This has its costs, but I think that there is a major compensating benefit, namely that the way the book is organized makes it clearer just what depends on what. It also reflects something of the historical order in which ideas emerged.

Many thanks are due to generations of students and to JC Beall, Hubie Chen, Torkel Franzén, Andy Fugard, Jeffrey Ketland, Jonathan Kleid, Fritz Mueller, Tristan Mills, Jeff Nye, Alex Paseau, Michael Potter, José F. Ruiz, Wolfgang Schwartz and Brock Sides for comments on draft chapters. Particular thanks to Richard Zach both for pointing out a number of mistakes, large and small, in an early draft and for suggestions that much improved the book. And particular thanks too to Seamus Holland, Luca Incurvati, Brian King, and Mary Leng, who read carefully through a late draft in a seminar together and made many helpful comments.

Like so many others, I am also hugely grateful to Donald Knuth, Leslie Lamport and the LaTeX community for the document processing tools which make typesetting a mathematical text like this one such a painless business.

1 What Gödel's Theorems say

1.1 Basic arithmetic

It seems to be child's play to grasp the fundamental notions involved in the arithmetic of addition and multiplication. Starting from zero, there is a sequence of 'counting' numbers, each having just one immediate successor. This sequence of numbers – officially, *the natural numbers* – continues without end, never circling back on itself; and there are no 'stray' numbers, lurking outside this sequence. Adding n to m is the operation of starting from m in the number sequence and moving n places along. Multiplying m by n is the operation of (starting from zero and) repeatedly adding m, n times. And that's about it.

Once these fundamental notions are in place, we can readily define many more arithmetical notions in terms of them. Thus, for any natural numbers m and n, $m < n$ if there is a number $k \neq 0$ such that $m + k = n$. m is a factor of n if $0 < m$ and there is some number k such that $0 < k$ and $m \times k = n$. m is even if it has 2 as a factor. m is prime if $1 < m$ and m's only factors are 1 and itself. And so on.

Using our basic and/or defined concepts, we can then make various general claims about the arithmetic of addition and multiplication. There are familiar elementary truths like 'addition is commutative', i.e. for any numbers m and n, we have $m + n = n + m$. There are also yet-to-be-proved conjectures like Goldbach's conjecture that every even number greater than two is the sum of two primes.

That second example illustrates the truism that it is one thing to understand what we'll call *the language of basic arithmetic* (i.e. the language of the addition and multiplication of natural numbers, together with the standard first-order logical apparatus), and it is another thing to be able to evaluate claims that can be framed in that language.

Still, it is extremely plausible to suppose that, whether the answers are readily available to us or not, questions posed in the language of basic arithmetic do *have* entirely determinate answers. The structure of the number sequence is (surely) simple and clear. There's a single, never-ending sequence, starting with zero; each number is followed by a unique successor; each number is reached by a finite number of steps from zero; there are no repetitions. The operations of addition and multiplication are again (surely) entirely determinate; their outcomes are fixed by the school-room rules. So what more could be needed to fix the truth or falsity of propositions that – perhaps via a chain of definitions – amount to claims of basic arithmetic? To put it fancifully: God sets down the number sequence and specifies how the operations of addition and multiplication work. He has then done all he needs to do to make it the case that Goldbach's conjecture is true (or false, as the case may be).

Of course, that last remark is *far* too fanciful for comfort. We may find it compelling to think that the sequence of natural numbers has a definite structure, and that the operations of addition and multiplication are entirely nailed down by the familiar rules. But what is the real content of the thought that the truth-values of all basic arithmetic propositions are thereby 'fixed'?

Here's one initially attractive way of giving non-metaphorical content to that thought. The idea is that we can specify a bundle of fundamental assumptions or *axioms* which somehow pin down the structure of the number sequence,[1] and which also characterize addition and multiplication (after all, it is entirely natural to suppose that we *can* give a reasonably simple list of true axioms to encapsulate the fundamental principles so readily grasped by the successful learner of school arithmetic). So suppose that φ is a proposition which can be formulated in the language of basic arithmetic. Then, the plausible suggestion continues, the assumed truth of our axioms always 'fixes' the truth-value of any such φ in the following sense: either φ is logically deducible from the axioms by a normal kind of proof, and so is true; or $\neg\varphi$ is deducible from the axioms, and so φ is false.[2] We may not, of course, actually stumble on a proof one way or the other. But the picture is that the axioms contain the information from which the truth-value of any basic arithmetical proposition can in principle be deductively extracted by deploying familiar step-by-step logical rules of inference.

Logicians say that a theory T is *(negation)-complete* if, for every sentence φ in the language of the theory, either φ or $\neg\varphi$ is deducible in T's proof system. So, put into this jargon, the suggestion we are considering is: we should be able to specify a reasonably simple bundle of true axioms which taken together give us a *complete* theory of basic arithmetic. In other words, we can find a theory in which we can prove in principle the truth or falsity of any claim about addition and/or multiplication (or at least, any claim we can state using quantifiers, connectives and identity). And if that's right, truth in basic arithmetic could just be equated with provability in some appropriate system.

It is tempting to say more. For what will the axioms of basic arithmetic look like? Here's a candidate: 'For every natural number, there's a unique next one'. And this claim looks very like a definitional triviality. You might say: it is just part of what we *mean* by talk of the natural numbers that we are dealing with an ordered sequence where each member of the sequence has a unique successor. And, plausibly, other candidate axioms are similarly true by definition (either bald definitions, or derivable from logic plus definitions).

But if both of those thoughts are right – if the truths of basic arithmetic

[1] There are issues lurking here about what counts as 'pinning down a structure' using a bunch of axioms: we'll have to return to some of these issues in due course.

[2] 'Normal proof' is vague, and later we will need to be more careful: but the idea is that we don't want to countenance, e.g., 'proofs' with an infinite number of steps.

all flow deductively from logic plus definitions – then true arithmetical claims would be simply *analytic* in the philosophers' sense (follows from logic plus definitions).[3] And this so-called 'logicist' view would then give us a very neat explanation of the special certainty and the necessary truth of correct claims of basic arithmetic.

1.2 Incompleteness

But now, in headline terms, *what Gödel's First Incompleteness Theorem shows is that that the entirely natural idea that we can axiomatize basic arithmetic is wrong.* Suppose we try to specify a suitable axiomatic theory T that seems to capture the structure of the natural number sequence and pin down addition and multiplication (and maybe a lot more besides). Then Gödel gives us a recipe for coming up with a corresponding sentence G_T, couched in the language of basic arithmetic, such that (i) we can show (on very modest assumptions) that neither G_T nor $\neg\mathsf{G}_T$ can be proved in T, and yet (ii) we can also recognize that G_T will be true so long as T is consistent.

This is surely astonishing. Somehow, it seems, the class of basic arithmetic truths about addition and multiplication will *always* elude our attempts to pin it down by a fixed set of fundamental assumptions (definitional or otherwise) from which we can deduce everything else.

How does Gödel show this in his great 1931 paper? Well, note how we can use numbers and numerical propositions to encode facts about all sorts of things (for a trivial example, students in the philosophy department might be numbered off in such a way that one student's code-number is less than another's if the first student is older than the second; a student's code-number is even if the student in question is female; and so on). In particular, then, we can use numbers and numerical propositions to encode facts about what can be proved in a theory T. And what Gödel did is find a general method that enabled him to take any theory T strong enough to capture a modest amount of basic arithmetic and construct a corresponding arithmetical sentence G_T which encodes the claim 'The sentence G_T itself is unprovable in theory T'.

Suppose that T is a *sound* theory of arithmetic, i.e. T has true axioms and a reliably truth-preserving deductive logic. Then everything T proves must be true. But if T were to prove its Gödel sentence G_T, then it would prove a falsehood (since what G_T 'says' would then be untrue). Hence, if T is sound, G_T is unprovable in T. But then G_T is *true* (since it correctly 'says' it is unprovable). Hence $\neg\mathsf{G}_T$ is false; and so that too can't be proved by T. In sum, still assuming T is sound, neither G_T nor its negation will be provable in T: therefore T can't

[3] Thus Gottlob Frege, writing in his wonderful *Foundations of Arithmetic*, urges us to seek the proof of a mathematical proposition by 'following it up right back to the primitive truths. If, in carrying out this process, we come only on general logical laws and on definitions, then the truth is an analytic one.' (1884/1950, p. 4)

be negation-complete. And in fact we don't even need to assume that T is sound: Gödel goes on to show that T's mere consistency is enough to guarantee that G_T is unprovable.

Our reasoning here about 'This sentence is unprovable' is of course reminiscent of the Liar paradox, i.e. the ancient conundrum about 'This sentence is false', which is false if it is true and true if it is false. You might well wonder whether Gödel's argument leads to a paradox rather than a theorem. But not so. As we will see, there is nothing at all suspect about Gödel's First Theorem as a technical result about formal axiomatized systems.

'Hold on! If we can locate G_T, a Gödel sentence for our favourite theory of arithmetic T, and can argue that G_T is true-but-unprovable, why can't we just patch things up by adding it to T as a new axiom?' Well, to be sure, if we start off with theory T (from which we can't deduce G_T), and add G_T as a new axiom, we'll get an expanded theory $U = T + \mathsf{G}_T$ from which we *can* quite trivially deduce G_T. But we can now just re-apply Gödel's method to our improved theory U to find a new true-but-unprovable-in-U arithmetic sentence G_U that encodes 'I am unprovable in U'. So U again is incomplete. Thus T is not only incomplete but, in a quite crucial sense, is *incompletable.*

Let's emphasize this key point. There's nothing mysterious about a theory failing to be negation-complete, plain and simple. Imagine the departmental administrator's 'theory' T which records some basic facts about the course selections of a group of students – the language of T, let's suppose, is very limited and can only be used to tell us about who takes what course in what room when. From the 'axioms' of T we'll be able, let's suppose, to deduce further facts such as that Jack and Jill take a course together, and at least ten people are taking the logic course. But if there's no axiom in T about their classmate Jo, we might not be able to deduce either J = 'Jo takes logic' or $\neg\mathsf{J}$ = 'Jo doesn't take logic'. In that case, T isn't yet a negation-complete story about the course selections of students. However, that's just boring: for the 'theory' about course selection is no doubt completable (i.e. it can be expanded to settle every question that can be posed in its very limited language). By contrast, what gives Gödel's First Theorem its real bite is that it shows that any properly axiomatized and consistent theory of basic arithmetic must *remain* incomplete, whatever our efforts to complete it by throwing further axioms into the mix.

Note, by the way, that since G_U can't be derived from U, i.e. $T + \mathsf{G}_T$, it can't be derived from the original T either. And we can keep on going: iteration of the same trick generates a never-ending stream of independent true-but-unprovable sentences for any nice axiomatized theory of basic arithmetic T.

1.3 More incompleteness

Incompletability doesn't just affect basic arithmetic. For the next simplest example, consider the mathematics of the rational numbers (fractions, positive and negative). This embeds basic arithmetic in the following sense. Take the positive rationals of the form $n/1$ (where n is a natural number). These of course form a sequence with the structure of the natural numbers. And the usual notions of addition and multiplication for rational numbers, when restricted to rationals of the form $n/1$, correspond exactly to addition and multiplication for the natural numbers. So suppose that there were a negation-complete axiomatic theory T of the rationals such that, for any proposition ψ of rational arithmetic, either ψ or $\neg\psi$ can be deduced from T. Then, in particular, given any proposition ψ' about the addition and/or multiplication of rationals of the form $n/1$, T will entail either ψ' or $\neg\psi'$. But then T plus simple instructions for rewriting such propositions ψ' as propositions about the natural numbers would be a negation-complete theory of basic arithmetic – which is impossible by the First Incompleteness Theorem. Hence there can be no complete theory of the rationals.

Likewise for any stronger theory that can define (an analogue of) the natural-number sequence. Take set theory for example. Start with the empty set $\varnothing$. Form the set $\{\varnothing\}$ containing $\varnothing$ as its sole member. Now form the set $\{\varnothing, \{\varnothing\}\}$ containing the empty set we started off with plus the set we've just constructed. Keep on going, at each stage forming the set of sets so far constructed (a legitimate procedure in any standard set theory). We get the sequence

$$\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}, \{\varnothing, \{\varnothing\}, \{\varnothing, \{\varnothing\}\}\}, \ldots$$

This has the structure of the natural numbers. It has a first member (corresponding to zero); each member has one and only one successor; it never repeats. We can go on to define analogues of addition and multiplication. If we could have a negation-complete axiomatized set theory, then we could, in particular, have a negation-complete theory of the fragment of set-theory which provides us with an analogue of arithmetic; and then adding a simple routine for translating the results for this fragment into the familiar language of basic arithmetic would give us a complete theory of arithmetic. So, by Gödel's First Incompleteness Theorem again, there cannot be a negation-complete set theory.

In sum, any axiomatized mathematical theory T rich enough to embed a reasonable amount of the basic arithmetic of the addition and multiplication of the natural numbers must be incomplete and incompletable – yet we can recognize certain 'Gödel sentences' for T to be not only unprovable but to be true if T is consistent.

1.4 Some implications?

Gödelian incompleteness immediately defeats what is otherwise a surely attractive suggestion about the status of arithmetic – namely the logicist idea that it flows deductively from definitional truths that articulate the very ideas of the natural numbers, addition and multiplication.

But then, how *do* we manage somehow to latch on to the nature of the unending number sequence and the operations of addition and multiplication in a way that outstrips whatever rules and principles can be captured in definitions? At this point it can seem that we must have a rule-transcending cognitive grasp of the numbers which underlies our ability to recognize certain 'Gödel sentences' as correct arithmetical propositions. And if you are tempted to think so, then you may well be further tempted to conclude that minds such as ours, capable of such rule-transcendence, can't be machines (supposing, reasonably enough, that the cognitive operations of anything properly called a machine can be fully captured by rules governing the machine's behaviour).

So there's apparently a quick route from reflections about Gödel's First Theorem to some conclusions about the nature of arithmetical truth and the nature of the minds that grasp it. Whether those conclusions really follow will emerge later. For the moment, we have an initial idea of what the Theorem says and why it might matter – enough, I hope, already to entice you to delve further into the story that unfolds in this book.

1.5 The unprovability of consistency

If T can prove even a modest amount of basic arithmetic, it will be able to prove $\mathsf{0} \neq \mathsf{1}$. So if T *also* proves $\mathsf{0} = \mathsf{1}$, it is inconsistent. Conversely, if T is inconsistent, then – since an inconsistent theory can prove anything[4] – it can prove $\mathsf{0} = \mathsf{1}$. However, we said that we can use numerical propositions to encode facts about what can be proved in T. So there will be in particular a *numerical* proposition Con_T that encodes the claim that T can't prove $\mathsf{0} = \mathsf{1}$, i.e. encodes in a natural way the claim that T is consistent.

We already know, however, that there is also a numerical proposition which encodes the claim that G_T is unprovable, namely G_T itself!

So this means that (part of) the conclusion of Gödel's First Theorem, namely the claim that if T is consistent, then G_T is unprovable, can itself be encoded by a numerical proposition, namely $\mathsf{Con}_T \to \mathsf{G}_T$.

Now for another wonderful Gödelian insight. It turns out that the informal reasoning that we use, outside T, to prove 'if T is consistent, then G_T is unprovable' is elementary enough to be mirrored by reasoning inside T (i.e. by reasoning with numerical propositions which encode facts about T-proofs). Or at least that's true so long as T satisfies conditions only slightly stronger than the First Theorem assumes. So, again on modest assumptions, we can show that T actually *proves* $\mathsf{Con}_T \to \mathsf{G}_T$.

But the First Theorem has already shown that if T is consistent it can't prove G_T. So it immediately follows that if T is consistent it can't prove Con_T. And *that*

[4] There are, to be sure, deviant non-classical logics in which this principle doesn't hold. In this book, however, we aren't going to take further note of them, if only because of considerations of space.

is Gödel's Second Incompleteness Theorem. Roughly interpreted: nice theories that include enough basic arithmetic can't prove their own consistency.[5]

1.6 More implications?

Suppose there's a genuine issue about whether T is consistent. If T *had* been able to prove itself consistent, would that have settled the matter and shown that it *is* consistent? Of course not. For if T were inconsistent we can derive anything within T – including a statement of its own consistency!

But then why does the Second Theorem matter? Think of it this way. If a nice arithmetical theory T can't even prove that it is *itself* consistent, it certainly can't prove that a *richer* theory T^+ is consistent (since if the richer theory is consistent, then any cut-down part of it is consistent). Thus we can't use nice 'finitistic' reasoning of the kind we can encode in ordinary arithmetic to prove other more 'risky' mathematical theories are in good shape. For example, we can't use unproblematic arithmetical reasoning to convince ourselves of the consistency of set theory (with its postulation of a universe of wildly infinite sets).

And *that* is a very interesting result, for it seems to sabotage what is called Hilbert's Programme, which is precisely the project of defending the wilder reaches of infinitistic mathematics by giving 'safe' consistency proofs. A lot more about this in due course.

1.7 What's next?

What we've said so far, of course, has been very sketchy and introductory. We must now start to do better – though for the next few chapters our discussions will remain fairly informal. In Chapter 2, we introduce the notions of effective computability, decidability and enumerability, notions we are going to need in what follows. Then in Chapter 3, we explain more carefully what we mean by talking about an 'axiomatized theory' and prove some elementary results about theories in general. In Chapter 4, we introduce some concepts relating specifically to axiomatized theories of arithmetic. Then in Chapter 5 we prove a neat and relatively easy result – namely that any so-called 'sufficiently strong' axiomatized theory of arithmetic is negation incomplete. For reasons that we'll explain, this informal result falls well short of Gödel's First Incompleteness Theorem. But it provides a very nice introduction to some key ideas that we'll be developing more formally in the ensuing chapters.

[5] That *is* rough. The Second Theorem shows that T can't prove Con_T which is certainly *one* entirely natural way of expressing T's consistency inside T. But could there be some *other* sentence of T, Con^*_T, which *also* in some good sense expresses T's consistency but which T *can* prove? We'll have to return to this sort of question later.

2 Decidability and enumerability

This chapter briskly introduces three related ideas that we'll need in the next few chapters. Later in the book, we'll return to these ideas and give a sharp technical treatment of them. But for present purposes, an informal, intuitive presentation is enough.

2.1 Effective computability, effective decidability

Familiar arithmetic routines (e.g. for squaring a number or finding the highest common factor of two numbers) give us ways of *effectively computing* an answer. Likewise other familiar routines (e.g. for testing whether a number is prime) give us ways of *effectively deciding* whether some property holds.

When we say such routines are *effective* we mean that (i) they involve entirely determinate, mechanical, step-by-step procedures. (ii) There isn't any room left for the exercise of imagination or intuition or fallible human judgement. (iii) To execute the procedures, we don't have to resort to outside oracles (or other sources of empirical information). (iv) We don't have to resort to random methods (coin tosses). And (v) the procedures are guaranteed to terminate and deliver a result after a finite number of steps.

In a word, effective procedures involve following an *algorithm* – i.e. following a series of step-by-step instructions (instructions which are pinned down in advance of their execution), with each small step clearly specified in every detail (leaving no room for doubt as to what does and what doesn't count as executing the step), and such that following the instructions will always deliver a result. Such algorithms can be executed by a dumb computer. Indeed it is natural to turn that last point into an informal definition:

> An algorithmic procedure is a *computable* one, i.e. one that a suitably programmed computer can execute and that is guaranteed to deliver a result in finite time.

And then we can give two further interrelated definitions:[1]

> A function is *effectively computable* iff there is an algorithmic procedure that a suitably programmed computer could use for calculating the value of the function for any given argument.

[1] For more about how to relate these two definitions via the notion of a 'characteristic function', see Section 9.6. We are assuming for the moment that functions are 'total' – i.e. defined for all arguments in the relevant domain. And 'iff' is, of course, the standard logicians' abbreviation for 'if and only if'.

> A property is *effectively decidable* iff there is an algorithmic procedure that a suitably programmed computer could use to decide whether the property applies in any given case.

But what kind of computer do we have in mind here when we say that an algorithmic procedure is one that a computer can execute? We need to say something here about the relevant computer's (a) *size and speed*, and (b) *architecture.*

(a) A real-life computer is limited in size and speed. There will be some upper bound on the size of the inputs it can handle; there will be an upper bound on the size of the set of instructions it can store; there will be an upper bound on the size of its working memory. And even if we feed in inputs and instructions it can handle, it is of little practical use to us if the computer won't finish doing its computation for centuries.

Still, we are going to cheerfully abstract from all those 'merely practical' considerations of size and speed. In other words, we will say that a function is effectively computable if there is a finite set of step-by-step instructions which a computer could in principle use to calculate the function's value for any particular arguments, given memory, working space and time enough. Likewise, we will say that a property is effectively decidable if there is a finite set of step-by-step instructions a computer can use which is in principle guaranteed to decide whether the property applies in any given case, again abstracting from worries about memory and time limitations. Let's be clear, then: 'effective' here does *not* mean that the computation must be feasible for us, on existing computers, in real time. So, for example, we count a numerical property as effectively decidable in this broad sense even if on existing computers it might take more time to compute whether a given number has it than we have left before the heat death of the universe. It is enough that there's an algorithm that works in theory and would deliver an answer in the end, if only we had the computational resources to use it and could wait long enough.

'But then,' you might well ask, 'why on earth bother with these radically idealized notions of computability and decidability. If we allow procedures that may not deliver a verdict in the lifetime of the universe, what good is that? If we are interested in issues of computability, shouldn't we really be concerned not with idealized-computability-in-principle but with some stronger notion of *practicable* computability?'

That's an entirely fair challenge. And modern computer science has much to say about grades of computational complexity and different levels of feasibility. However, we will stick to our idealized notions of computability and decidability. Why? Because we'll later be proving a range of limitative theorems, about what can't be algorithmically decided. And by working with a weak 'in principle' notion of what is required for being decidable, our impossibility results will be exceedingly strong – they won't depend on mere contingencies about what is practicable, given the current state of our software and hardware, and given real-world limitations of time or resources. They show, in particular, that some problems are not mechanically decidable, even on the most generous understanding of that idea.

(b) We've said that we are going to be abstracting from limitations on storage etc. But you might suspect that this still leaves much to be settled. Doesn't the 'architecture' of a computing device affect what it can compute?

The short answer is 'no'. And intriguingly, some of the central theoretical questions here were the subject of intensive investigation even before the first electronic computers were built. Thus, in the mid 1930s, Alan Turing famously analysed what it is for a numerical function to be step-by-step computable in terms of the capacities of a *Turing machine* (a computer following a program built up from extremely simple steps: for explanations and examples, see Chapter **??**). Now, it is easy to spin variations on the details of Turing's original story. For example, a standard Mark I Turing machine has just a single 'tape' or workspace to be used for both storing and manipulating data: but we can readily describe a Mark II machine which has (say) two tapes – one to be used as a main workspace, and a separate one for storing data. Or we can consider a computer with unlimited 'Random Access Memory' – that is to say, an idealized version of a modern computer with an unlimited set of registers in which it can store various items of working data ready to be retrieved into its workspace when needed.[2] The details don't matter here and now. What does matter is that exactly the same functions are computable by Mark I Turing machines, Mark II machines, and by register machines, despite their different architectures. Indeed, *all* the definitions of algorithmic computability by idealized computers that have ever been seriously proposed turn out to be equivalent. In a slogan, *algorithmic computability is architecture independent.* Likewise, what is algorithmically decidable is architecture independent.

Let's put that a bit more carefully, in two stages. First, there's a Big Mathematical Result – or rather, a cluster of results – that can conclusively be proved about the equivalence of various definitions of computation for *numerical* functions and properties. And this Big Mathematical Result supports the claim Turing famously makes in his classic paper published in 1936, which we can naturally call

> *Turing's Thesis* The numerical functions that are computable in the intuitive sense are just those functions that are computable by a Turing machine. Likewise, the numerical questions that are effectively decidable in the intuitive sense are just those questions that are decidable by a suitable Turing machine.

This claim – we'll further explore its content in Chapter **??** – correlates an intuitive notion with a sharp technical analysis. So you might perhaps think it is not the sort of thing for which we can give a proof (though we will challenge

[2] The theoretical treatment of unlimited register machines was first given in (Shepherdson and Sturgis, 1963); there is a very accessible presentation in the excellent (Cutland, 1980).

that view in Chapter **??**). But be that as it may. This much, at any rate is true: after some seventy years, no successful challenge to Turing's Thesis has been mounted. Which means that we can continue to talk informally about intuitively computable numerical functions and effectively decidable properties of numbers, and be very confident that we are referring to fully determinate classes.

But now, second, what about the idea of being computable as applied to non-numerical functions (like truth-functions) or the idea of being effectively decidable as applied to non-numerical properties (like the property of being an axiom of some theory)? Are these ideas determinate too?

Well, think how a real-world computer can be used to evaluate a truth-function or decide whether a formal expression is an axiom in a given system. In the first case, we code the truth-values *true* and *false* using numbers, say 0 and 1, and then do a numerical computation. In the second case, we write a program for manipulating strings of symbols, and again – though this time behind the scenes – these strings get correlated with binary codes, and it is these numbers that the computer works on. In the end, using numerical codings, the computations in both cases are done on numbers after all.

Now generalize that thought. A natural suggestion is that *any* computation dealing with sufficiently determinate and distinguishable Xs can be turned into an equivalent numerical computation via the trick of using simple numerical codes for the different Xs. More carefully: by a relatively trivial algorithm, we can map Xs to numbers; we can then do the appropriate core computation on the numbers; and then another trivial algorithm translates the result back into a claim about X's.

Fortunately, we don't need to assess that natural suggestion in its full generality. For the purposes of this book, the non-numerical computations we are interested in are cases where the Xs are expressions from standard formal languages, or sequences of expressions, etc. And in those cases, there's no doubt at all that we can algorithmically map claims about such things to corresponding claims about numbers (see Sections 3.5, 12.1, 12.2). So the question e.g. whether a certain property of formulae is a decidable one can be translated quite uncontentiously into the question whether a corresponding numerical property is a decidable one. Given Turing's Thesis that it is quite determinate what counts as a decidable property of *numbers*, it follows that it is quite determinate what counts as a decidable property of *formal expressions*. And similarly for other cases we are interested in.

2.2 Enumerable sets

Having introduced the twin ideas of effective computability and decidability, we go on to explain the related notion of effective enumerability. But before we can do that in the next section, we need the prior notion of (plain) enumerability.

Suppose, then, that Σ is some set of items: its members might be numbers, strings of symbols, proofs, sets or whatever. We say that Σ is *enumerable* if its members can – at least in principle – be listed off (the zero-th, first, second, . . .) with every member appearing on the list; repetitions are allowed, and the list may be infinite. It is tidiest to think of the empty set as the limiting case of an enumerable set: after all, it is enumerated by the empty list!

That informal definition will serve well enough. But, for the pernickety, we can make it more rigorous in various equivalent ways. Here, we'll give just one. And to do this, let's introduce some standard jargon and notation that we'll need later anyway (for the moment, we'll focus on one-place functions).

i. A function maps arguments in some *domain* to unique values. Suppose the function f is defined for *all* arguments in the domain Δ; and suppose that the values of f all belong to the set Γ. Then we write

   $$f\colon \Delta \to \Gamma$$

   and say that f is a (total) function from Δ *into* Γ.

ii. The *range* of a function $f\colon \Delta \to \Gamma$ is the set $\{f(x) \mid x \in \Delta\}$: in other words, it is the set of $y \in \Gamma$ such that f maps some $x \in \Delta$ to y.

iii. A function $f\colon \Delta \to \Gamma$ is *surjective* iff the range of f is the whole of Γ – i.e. if for every $y \in \Gamma$ there is some $x \in \Delta$ such that $f(x) = y$. (If you prefer that in English, you can say that such a function is '*onto*', since it maps Δ onto the whole of Γ.)

iv. We use '$\mathbb{N}$' to denote the set of all natural numbers.

Then here's our first official definition:

> The set Σ is *enumerable* iff it is either empty or there is a surjective function $f\colon \mathbb{N} \to \Sigma$. (We can say that such a function enumerates Σ.)

To see that this comes to the same as our original informal definition, just note the following two points. (a) Suppose we have a list of all the members of Σ in some order (starting with the zero-th, and perhaps an infinite list, perhaps with repetitions). Then take the function f defined as follows $f(n) = n$-th member of the list, if the list goes on that far, or $f(n) = f(0)$ otherwise. Then f is a surjection $f\colon \mathbb{N} \to \Sigma$. (b) Suppose conversely that f is surjection $f\colon \mathbb{N} \to \Sigma$. Then, if we successively evaluate f for the arguments 0, 1, 2, . . . , we get a list of values $f(0), f(1), f(2) \ldots$ which by hypothesis contains all the elements of Σ, with repetitions allowed.

Here's a quick initial result: If two sets are enumerable, so is the result of combining their members into a single set. (Or if you prefer that in symbols: if Σ_1 and Σ_2 are enumerable so is $\Sigma_1 \cup \Sigma_2$.)

Proof Suppose there is a list of members of Σ_1 and a list of members of Σ_2. Then we can interleave these lists by taking members of the two sets alternately, and the result will be a list of the union of those two sets. (More formally, suppose f_1 enumerates Σ_1 and f_2 enumerates Σ_2. Put $g(2n) = f_1(n)$ and $g(2n+1) = f_2(n)$; then g enumerates $\Sigma_1 \cup \Sigma_2$.) $\boxtimes$

That was easy and trivial. Here's another much more important result – famously proved by Georg Cantor[3] – which is also easy, but certainly not trivial:

Theorem 1 *There are infinite sets that are not enumerable.*

Proof Consider the set $\mathbb{B}$ of infinite binary strings (i.e. the set of unending strings like '0110001010011...'). There's obviously an infinite number of them. Suppose, for reductio, that we could list off these strings in some order. More carefully, suppose that there is an enumerating function which maps the natural numbers onto the binary strings as follows:

$$\begin{array}{ccl}
0 & \to & \underline{0}110001010011\ldots \\
1 & \to & 1\underline{1}00101001101\ldots \\
2 & \to & 11\underline{0}0101100001\ldots \\
3 & \to & 000\underline{1}111010101\ldots \\
4 & \to & 1101\underline{1}11011101\ldots \\
\ldots & \ldots &
\end{array}$$

Read off down the diagonal, taking the n-th digit of the n-th string (in our example, this produces 01011...). Now flip each digit, swapping 0s and 1s (in our example, yielding 10100...). By construction, this 'flipped diagonal' string differs from the initial string on our original list in the first place, differs from the next string in the second place, and so on. So our diagonal construction defines a string that isn't on the list, contradicting the assumption that our enumerating function is 'onto', i.e. that our list contains *all* the binary strings. So $\mathbb{B}$ is infinite, but not enumerable. $\boxtimes$

It's worth pausing to add three quick comments about this result for later use.

a. An infinite binary string $b_0b_1b_2\ldots$ can be thought of as characterizing a real number $0 \leq b \leq 1$ in binary digits. So our theorem shows that the real numbers between in the interval $[0,1]$ can't be enumerated (and hence we can't enumerate *all* the reals either).

b. An infinite binary string $b_0b_1b_2\ldots$ can also be thought of as characterizing a corresponding set of natural numbers Σ, where $n \in \Sigma$ just if $b_n = 0$. So our theorem is equivalent to the result that the set of *sets* of natural numbers can't be enumerated.

[3] Cantor first established this key result in his (1874), using, in effect, the Bolzano-Weierstrass theorem. The neater 'diagonal argument' we give here first appears in his (1891).

c. A third way of thinking of an infinite binary string $b_0b_1b_2\ldots$ is as characterizing a corresponding function f, i.e. the function that maps each natural number to one of the numbers $\{0,1\}$, where $f(n) = b_n$. So our theorem is also equivalent to the result that the set of *functions* from the natural numbers to $\{0,1\}$ can't be enumerated. (Put in terms of functions, the trick in the proof is to suppose that these functions *can* be enumerated in a list $f_0, f_1, f_2, \ldots$, define another function by 'going down the diagonal and flipping digits', i.e. define $d(n) = 1 - f_n(n)$, and then note that this diagonal function can't be on the list after all. We'll soon meet this version of the 'diagonalization' trick again.)

It is also worth noting that non-enumerable sets have to be, in a good sense, a *lot* bigger than enumerable ones. Suppose Σ is a non-enumerable set; suppose $\Delta \subset \Sigma$ is some enumerable subset of Σ; and let $\Gamma = \Sigma \setminus \Delta$ be the set you get by removing the members of Δ from Σ. Then this difference set will *still* be non-emumerably infinite – for if it were enumerable, $\Sigma = \Delta \cup \Gamma$ would be enumerable after all (by the easy result we proved above).

2.3 Effective enumerability

Note carefully: to say that a set is enumerable is *not* to say that we can produce a 'nice' algorithmically computable function that does the enumeration. The claim is only that there is *some* function or other that does the job. So let's now add a further official definition:

> The set Σ is *effectively* enumerable iff it is either empty or there is an *effectively computable* function that enumerates it.[4]

In other words, a set is effectively enumerable if an (idealized) computer could be programmed to start producing a list of its members such that any member will be eventually mentioned – the list may have no end, and may contain repetitions, so long as any item in the set eventually appears.

It is often crucial whether a set can be effectively enumerated in this sense. A *finite* set of finitely specifiable objects is always effectively enumerable: any listing will do, and – since it is finite – it could be stored in an idealized computer and spat out on demand. And for a simple example of an effectively enumerable *infinite* set, imagine an algorithm that generates the natural numbers one at a time in order, ignores those that fail the well-known (mechanical) test for being prime, and lists the rest: this procedure generates a never-ending list on which every prime will eventually appear – so the primes are effectively enumerable.

We'll see later that there are key examples of infinite sets which are enumerable but which *can't* be effectively enumerated.

[4] Jargon alert! Terminology hereabouts isn't stable: some writers use 'enumerable' to mean *effectively* enumerable, and use e.g. 'denumerable' for our wider notion.

3 Axiomatized formal theories

Gödel's Incompleteness Theorems tell us about the limits of axiomatized theories of arithmetic. Or rather, more carefully, they tell us about the limits of axiomatized *formal* theories of arithmetic. But what exactly does this mean? This chapter starts by exploring the idea. We then move on to prove some elementary but rather important general results about axiomatized formal theories.

3.1 Formalization as an ideal

Rather than just dive into a series of definitions, it is well worth pausing to remind ourselves of why we *care* about formalized theories.

Let's get back to basics. In elementary logic classes, we are drilled in translating arguments into an appropriate formal language and then constructing formal deductions of putative conclusions from given premisses. Why bother with formal languages? Because everyday language is replete with redundancies and ambiguities, not to mention sentences which simply lack clear truth-conditions. So, in assessing complex arguments, it helps to regiment them into a suitable artificial language which is expressly designed to be free from obscurities, and where surface form reveals logical structure.

Why bother with formal deductions? Because everyday arguments often involve suppressed premisses and inferential fallacies. It is only too easy to cheat. Setting out arguments as formal deductions in one style or another enforces honesty: we have to keep a tally of the premisses we invoke, and of exactly what inferential moves we are using. And honesty is the best policy. For suppose things go well with a particular formal deduction. Suppose we get from the given premisses to some target conclusion by small inference steps each one of which is obviously valid (no suppressed premisses are smuggled in, and there are no suspect inferential moves). Our honest toil then buys us the right to confidence that our premisses really do entail the desired conclusion.

Granted, outside the logic classroom we almost never set out deductive arguments in a fully formalized version. No matter. We have glimpsed a first ideal – arguments presented in an entirely perspicuous language with maximal clarity and with everything entirely open and above board, leaving no room for misunderstanding, and with all the arguments' commitments systematically and frankly acknowledged.[1]

[1] For an early and very clear statement of this ideal, see Frege (1882), where he explains the point of the first recognizably modern formal system of logic, presented in his *Begriffsschrift* (i.e. *Conceptual Notation*) of 1879.

Old-fashioned presentations of Euclidean geometry illustrate the pursuit of a related second ideal – the (informal) axiomatized theory. Like beginning logic students, school students used to be drilled in providing deductions, though the deductions were framed in ordinary geometric language. The game is to establish a whole body of theorems about (say) triangles inscribed in circles, by deriving them from simpler results which had earlier been derived from still simpler theorems that could ultimately be established by appeal to some small stock of fundamental principles or *axioms*. And the aim of this enterprise? By setting out the derivations of our various theorems in a laborious step-by-step style – where each small move is warranted by simple inferences from propositions that have already been proved – we develop a unified body of results that we can be confident must hold if the initial Euclidean axioms are true.

On the surface, school geometry perhaps doesn't seem very deep: yet making all its fundamental assumptions fully explicit is surprisingly difficult. And giving a set of axioms invites further enquiry into what might happen if we tinker with these assumptions in various ways – leading, as is now familiar, to investigations of non-Euclidean geometries.

Many other mathematical theories are also characteristically presented axiomatically.[2] For example, set theories are presented by laying down some basic axioms and exploring their deductive consequences. We want to discover exactly what is guaranteed by the fundamental principles embodied in the axioms. And we are again interested in exploring what happens if we change the axioms and construct alternative set theories – e.g. what happens if we drop the 'axiom of choice' or add 'large cardinal' axioms?

Now, even the most tough-minded mathematics texts which explore axiomatized theories are typically written in an informal mix of ordinary language and mathematical symbolism. Proofs are rarely spelt out in every formal detail, and their presentation falls short of the logical ideal of full formalization. But we will hope that nothing stands in the way of our more informally presented mathematical proofs being sharpened up into fully formalized ones – i.e. we hope that they *could* be set out in a strictly regimented formal language of the kind that logicians describe, with absolutely every inferential move made fully explicit and checked as being in accord with some overtly acknowledged rule of inference, with all the proofs ultimately starting from our explicitly given axioms. True, the extra effort of laying out everything in this kind of detail will almost never be worth the cost in time and ink. In mathematical practice we use enough formalization to convince ourselves that our results don't depend on illicit smuggled premisses or on dubious inference moves, and leave it at that – our motto is 'sufficient unto the day is the rigour thereof'.[3] But still, it *is* absolutely essential for good mathematics to achieve maximum precision and to

[2] For a classic defence, extolling the axiomatic method in mathematics, see Hilbert (1918).

[3] 'Most mathematical investigation is concerned not with the analysis of the complete process of reasoning, but with the presentation of such an abstract of the proof as is sufficient to convince a properly instructed mind.' (Russell and Whitehead, 1910–13, vol. 1, p. 3)

avoid the use of unexamined inference rules or unacknowledged assumptions. So, putting together the logician's aim of perfect clarity and honest inference with the mathematician's project of regimenting a theory into a tidily axiomatized form, we can see the point of the notion of an *axiomatized formal theory* as a composite ideal.

To forestall misunderstanding, let's stress that it isn't being supposed that we ought always be aiming to work inside axiomatized formal theories. Mathematics is hard enough even when done using the usual strategy of employing just as much rigour as seems appropriate to the case in hand.[4] And in any case, as mathematicians (and some philosophical commentators) are apt to stress, there is a lot more to mathematical practice than striving towards the logical ideal. For a start, we typically aim not merely for formal correctness but for *explanatory* proofs, which not only show that some proposition must be true, but in some sense make it clear *why* it is true. However, such points don't affect *our* point, which is that the business of formalization just takes to the limit features that we expect to find in good proofs anyway, i.e. precise clarity and lack of inferential gaps.

3.2 Formalized languages

So, putting together the ideal of formal precision and the ideal of regimentation into an axiomatic system, we have arrived at the concept of an axiomatized formal theory, which comprises a formalized *language*, a set of sentences from the language which we treat as *axioms* for the theory, and a *deductive system* for proof-building, so that we can derive theorems from the axioms.

In this section, we'll say just a bit more about the idea of a properly formalized language – though we'll be very brisk, as we don't want to get bogged down in details yet. Our main concern here is to emphasize a point about decidability.

Note that we are normally interested in *interpreted* languages – i.e. we are normally concerned not merely with patterns of symbols but with expressions which have an intended significance. After all, our formalized proofs are supposed to be just that, i.e. proofs with content, which show things to be true. True, we'll often be very interested in features of proofs that can be assessed independently of their significance (for example, we'll want to know whether a putative proof does obey the formal syntactic rules of a given deductive system). But it is one thing to ignore their semantics for some purposes; it is another thing entirely to drain formal proofs of all semantic significance.

Anyway, we can usefully think of a formal language L as in general being a pair $\langle \mathcal{L}, \mathcal{I} \rangle$, where the $\mathcal{L}$ is a syntactically defined system of expressions and $\mathcal{I}$

[4] See Lakatos (1976) for a wonderful exploration of how mathematics evolves. This gives some real sense of how regimenting proofs in order to clarify their assumptions – the process which formalization idealizes – is just one phase in the complex process that leads to the growth of mathematical knowledge.

gives the intended interpretation of these expressions.

(a) Start with the syntactic component $\mathcal{L}$. We'll assume that this is based on a finite alphabet of symbols.[5] We then first need to settle which symbols or strings of symbols make up $\mathcal{L}$'s logical vocabulary – typically this will comprise variables, symbols for connectives and quantifiers, the identity sign, and bracketing devices. Then we need similarly to specify which symbols or strings of symbols make up $\mathcal{L}$'s non-logical vocabulary, e.g. individual constants (names), predicates, and function-expressions. Finally, we need further syntactic construction rules to determine which finite sequences of logical and non-logical vocabulary constitute the well-formed formulae of $\mathcal{L}$ – its *wffs*, for short.

Some familiar ways of setting up the construction rules allow wffs which aren't *sentences*, where a sentence is a closed wff without any unquantified variables left dangling free. But note, it is only sentences which will be interpreted as expressing complete propositions that might be true or false.

All this should be very familiar from elementary logic: so just one comment on syntax. Given that the whole point of using a formalized language is to make everything as clear and determinate as possible, we don't want it to be a disputable matter whether a given sign or cluster of signs is e.g. a constant symbol or one-place predicate symbol of a given system $\mathcal{L}$. And, crucially, we don't want disputes either about whether a given string of symbols is an $\mathcal{L}$-wff or about whether a given wff is an $\mathcal{L}$-sentence.

So, whatever the details, for a properly formalized language, there should be clear and objective procedures, agreed on all sides, for *effectively deciding* whether a putative constant-symbol really is a constant, etc. Likewise we need to be able to effectively decide whether a string of symbols is a wff and also decide whether a wff is a sentence.

(b) Let's move on, then, to the interpretation $\mathcal{I}$. We'd like this to fix the content of each $\mathcal{L}$-sentence – and standardly, we fix the content of formal sentences by giving truth-conditions, i.e. by saying what it would take for a given sentence to be true. However, we can't, in the general case, do this just by giving a list, associating $\mathcal{L}$-sentences with truth-conditions (for the simple reason that there will be an unlimited number of sentences). We'll therefore normally aim for a 'compositional semantics', which tells us how to systematically work out the truth-condition of any $\mathcal{L}$-sentence in terms of the semantic significance of its syntactic parts.

What does such a compositional semantics look like? Here's a very quick reminder of one sort of case: again we'll assume that this is all broadly familiar from elementary logic. Suppose, then, that $\mathcal{L}$ has the usual syntax of the simplest predicate language (without identity or function symbols). A standard interpretation $\mathcal{I}$ will start by assigning *values* to constants and give *satisfaction conditions* for predicates. Thus, perhaps,

[5] We can always construct e.g. an unending supply of variables from a finite base by standard tricks like using repeated primes (to yield 'x', 'x′', 'x″', etc.).

> The value of ‘a’ is Socrates;
> the value of ‘b’ is Plato;
> something satisfies ‘F’ iff it is wise;
> an ordered pair of things satisfies ‘L’ iff the first loves the second.

Then $\mathcal{I}$ continues by giving us the obvious rules for assigning truth-conditions to atomic sentences, so that e.g. ‘Fa’ is true just in case the value of ‘a’ satisfies ‘F’ (i.e. iff Socrates is wise); ‘Lab’ is true just in case the ordered pair ⟨value of ‘a’, value of ‘b’⟩ satisfies ‘L’ (i.e. iff Socrates loves Plato); and so on.

Next, there are the usual rules for assigning truth-conditions to sentences built up out of simpler ones using the propositional connectives. And finally (the tricky bit!) we have to deal with the quantifiers. Take the existential case. Intuitively, if the quantifier is to range over people, then ‘∃xFx’ is true just if there is someone we could temporarily dub with the new name ‘c’ who would make ‘Fc’ come out true (because *that* person *is* wise). So let’s generalize this thought. To fix the truth-condition for quantified sentences on interpretation $\mathcal{I}$, we must specify a *domain* for quantifiers to run over (the set of people, for example), and then we can say that a sentence of the form ‘$\exists\nu\varphi(\nu)$’ is true on $\mathcal{I}$ just if, when we expand $\mathcal{L}$ with a new constant ‘c’, we can expand the interpretation $\mathcal{I}$ to deal with ‘c’ by giving it a value in the domain in such a way that ‘$\varphi(\mathsf{c})$’ is true on the expanded interpretation.[6] Similarly for universal quantifiers.

For the moment, let’s just make one comment about this kind of semantics, parallel to our comment about syntax. Given the aims of formalization, a compositional semantics needs to yield a single unambiguous interpretation for each sentence. Working out this interpretation should be a mechanical matter that doesn’t require any ingenuity or insight – i.e. it should again be effectively decidable what the interpretation is.

And of course, the usual accounts of the syntax and semantics of standard formal languages of logic have the decidability feature.

[6] Here, of course, ‘$\varphi(\mathsf{c})$’ stands in for any sentence with one or more occurrences of ‘c’; and ‘$\varphi(\nu)$’ is the result of replacing each occurrence of ‘c’ with the variable ν.

To connect our style of semantics to one that you might be more familiar with, note that something satisfies ‘F’ according to $\mathcal{I}$ iff it is in the set of wise people. Call that set associated with ‘F’ its *extension*. Then ‘Fa’ is true on interpretation $\mathcal{I}$ iff the value of ‘a’ is in the extension of ‘F’. Pursuing this idea, we can give a basically equivalent semantic story that deals with one-place predicates by assigning them subsets of the domain as extensions rather than by giving satisfaction conditions (similarly two-place predicates will be assigned sets of ordered pairs of elements of the domain, and so forth). Which is the way logic texts more usually tell the official semantic story, and for a very good reason. In logic, we are interested in finding the valid inferences, i.e. those which are such that, on *any* possible interpretation of the relevant sentences, if the premisses are true, the conclusion is true. Logicians therefore need to be able to *generalize* about all possible interpretations. Describing interpretations set-theoretically gives us a mathematically clean way of doing this generalizing work. However, in specifying a *particular* interpretation $\mathcal{I}$ for a given $\mathcal{L}$ we don’t need to put it in such overly set-theoretic terms. So we won’t.

3.3 Axiomatized formal theories

Now for the idea of an axiomatized formal theory, built in a formalized language (normally, of course, an interpreted formalized language). Again, it is issues about decidability which need to be highlighted.

(a) First, some wffs of our theory's language are to be selected as *axioms*, i.e. as fundamental assumptions of our theory (and we can take it without significant loss of generality that these are always sentences, i.e. closed wffs).

Since the fundamental aim of the axiomatization game is to see what follows from a bunch of axioms, we certainly don't want it to be a matter for dispute whether a given proof does or doesn't appeal only to axioms in the chosen set. Given a purported proof of some result, there should be an absolutely clear procedure for settling whether the input premisses are genuinely instances of the official axioms. In other words, for an axiomatized formal theory, we must be able to effectively decide whether a given wff is an axiom or not.

That doesn't, by the way, rule out theories with infinitely many axioms. We might want to say 'every wff of such-and-such a form is an axiom' (where there is an unlimited number of instances): that's permissible so long as it is still effectively decidable what counts as an instance of that form.

(b) Next, an axiomatized formal theory needs some deductive apparatus, i.e. some sort of formal *proof-system*. And we'll take proofs always to be *finite* arrays of wffs, arrays which are built up in ways that conform to the rules of the relevant proof-system, and whose only initial assumptions belong to the set of axioms.[7]

We'll take it that the core idea of a proof-system is once more very familiar from elementary logic. The differences between various equivalent systems of proof presentation – e.g. old-style linear proof systems vs. different styles of natural deduction proofs vs. tableau (or 'tree') proofs – don't matter for our purposes. What will matter is the strength of the system of rules we adopt. We will predominantly be working with some version of standard first-order logic with identity: but whatever system we adopt, it is crucial that we fix on a set of rules which enable us to settle, without room for dispute, what counts as a well-formed derivation in this system. In other words, we require the property of being a well-formed proof from premisses $\varphi_1, \varphi_2, \ldots, \varphi_n$ to conclusion

[7] We are not going to put any finite upper bound on the permissible length of proofs. So you might well ask: why not allow infinite arrays to count as proofs too? And indeed, there is some interest in theorizing about infinite 'proofs'. For example, we'll later consider the so-called ω-rule, which says that from the infinite array of premisses $\varphi(0)$, $\varphi(1)$, $\varphi(2)$, …, $\varphi(n)$, … we can infer $\forall x\varphi(x)$ when the quantifier runs over all natural numbers. But do note that finite minds can't really take in the infinite number of separate premisses in an application of the ω-rule: if we momentarily think we can, it's because we are confusing that impossible task with e.g. the finite task of taking in the premisses $\varphi(0)$, $\varphi(1)$, $\varphi(2)$, …, $\varphi(n)$ plus the premiss $(\forall x > n)\varphi(x)$. Hence, in so far as the business of formalization is primarily concerned to regiment and formalize the practices of ordinary mathematicians, albeit in an idealized way, it's natural at least to start by restricting ourselves to finite proofs of the general type we can cope with, even if we don't put any contingent bound on the length of proofs.

ψ in the theory's proof-system to be an effectively decidable one. The whole point of formalizing proofs is to set out the deductive structure of an argument with absolute determinacy, so we don't want it to be a disputable or subjective question whether the inference moves in a putative proof do or do not conform to the rules for proof-building for the formal system in use. Hence there should be a clear and effective procedure for deciding whether an array counts as a well-constructed derivation according to the relevant proof-system.[8]

Be careful! The claim here is only that it should be decidable whether an array of wffs presented as a well-constructed derivation really *is* a proper proof. This is *not* to say that we can always decide in advance whether a proof from given axioms exists to be discovered. Even in familiar first-order quantificational logic, for example, it is not in general decidable whether there exists a proof from certain premisses to a given conclusion (we'll be proving this undecidability result later, in Section **??**).

To summarize then, here again are the key headlines:

> T is an (interpreted) axiomatized formal theory just if (a) T is couched in an (interpreted) formalized language $\langle \mathcal{L}, \mathcal{I} \rangle$, such that it is effectively decidable what counts as a wff of $\mathcal{L}$, what counts as a sentence, what the truth-condition of any sentence is, etc., (b) it is effectively decidable which $\mathcal{L}$-wffs are axioms of T, and (c) T uses a proof-system such that it is effectively decidable whether an array of $\mathcal{L}$-wffs counts as conforming to the proof-building rules.

3.4 More definitions

Here are five more definitions specifically to do with theories:

i. Given a proof of the sentence (i.e. closed wff) φ from the axioms of the theory T using the background logical proof system, we will say that φ is a *theorem* of the theory. Using the standard abbreviatory symbol, we write: $T \vdash \varphi$.

ii. A theory T is *decidable* iff the property of being a theorem of T is an effectively decidable property – i.e. iff there is a mechanical procedure for determining, for any given sentence φ of the language of theory T, whether $T \vdash \varphi$.

iii. Assume now that T has a standard negation connective '$\neg$'. A theory T *decides* the wff φ iff either $T \vdash \varphi$ or $T \vdash \neg\varphi$. A theory T *correctly decides*

[8] When did the idea clearly emerge that properties like being a wff or an axiom or a proof *ought* to be decidable? It was arguably already implicit in Hilbert's conception of rigorous proof. But Richard Zach has suggested that an early source for the *explicit* deployment of the idea is von Neumann (1927).

φ just when, if φ is true (on the interpretation built into T's language), $T \vdash \varphi$, and if φ is false, $T \vdash \neg\varphi$.

iv. A theory T is *negation complete* iff T decides every sentence φ of its language (i.e. for every φ, either $T \vdash \varphi$ or $T \vdash \neg\varphi$).

v. T is *inconsistent* iff it proves some pair of theorems of the form φ, $\neg\varphi$.

Here's a very elementary example to illustrate some of these definitions. Consider a trivial pair of theories, T_1 and T_2, whose shared language consists of the propositional atoms 'p', 'q', 'r' and all the wffs that can be constructed out of them using the familiar propositional connectives, whose shared underlying logic is a standard natural deduction system for propositional logic, and whose sets of axioms are respectively {'¬p'} and {'p', 'q', '¬r'}. Given appropriate interpretations for the atoms, T_1 and T_2 are then both axiomatized formal theories. For it is mechanically decidable what is a wff of the theory, and whether a purported proof is indeed a proof from the given axioms. Both theories are consistent. Both theories are decidable; just use the truth-table test to determine whether a candidate theorem really follows from the axioms.

However, note that although T_1 is a decidable theory that doesn't mean T_1 decides every wff; it doesn't decide e.g. the wff '(q ∧ r)', since T_1's sole axiom doesn't entail either '(q ∧ r)' or '¬(q ∧ r)'. To stress the point: it is one thing to have a general way of mechanically deciding what is a theorem; it is another thing for a theory to be negation complete, i.e. to have the resources to prove or disprove every wff. But unlike T_1, T_2 *is* negation complete (any wff constructed from the three atoms using the truth-functional connectives has its truth-value decided, and the true ones can be proved and the false ones disproved).

This mini-example illustrates another crucial terminological point. You will be familiar with the idea of a deductive system being '(semantically) complete' or 'complete with respect to its standard semantics'. For example, a natural deduction system for propositional logic is said to be semantically complete when every inference which is semantically valid (i.e. truth-table valid) can be shown to be valid by a proof in the deductive system. But a theory's having a semantically complete logic is one thing, being a negation complete theory is something else entirely. For example, T_1 by hypothesis has a complete truth-functional *logic*, but is not a complete *theory.* For a more interesting example, we'll soon meet a formal arithmetic which we label 'Q'. This theory uses a standard quantificational deductive logic, which again is a (semantically) complete *logic*: but we can easily show that Q is not a (negation) complete *theory.*[9]

[9] Putting it symbolically may help. To say that a theory T with the set of axioms Σ is (negation) complete is to say that

for any sentence φ, either $\Sigma \vdash \varphi$ or $\Sigma \vdash \neg\varphi$;

while to say that a logic is (semantically) complete is to say that

for any set of sentences Γ, and any φ, if $\Gamma \vDash \varphi$ then $\Gamma \vdash \varphi$, Do watch out for this annoying and potentially dangerous double use of the term 'complete'; beware too of the use of 'decidable' and 'decides' for two significantly different ideas. These dual usages are unfortunately now entirely entrenched: you just have to learn to live with them.

3.5 The effective enumerability of theorems

Deploying our notion of effective enumerability, we can now state and prove the following portmanteau theorem (the last claim is the crucial part):

> **Theorem 2** *If T is an axiomatized formal theory then (i) the set of wffs of T, (ii) the set of sentences of T, (iii) the set of proofs constructible in T, and (iv) the set of theorems of T, can each be effectively enumerated.*

Proof sketch for (i) By hypothesis, T has a formalized language with a finite basic alphabet; and we can give an algorithm for mechanically enumerating all the possible finite strings of symbols formed from a finite alphabet. For example, start by listing all the strings of length 1, followed by all those of length 2 in some 'alphabetical order', followed by those of length 3 and so on. By the definition of a formalized language, there is a mechanical procedure for deciding which of these symbol strings count as wffs. So, putting these procedures together, as we ploddingly generate the possible strings we can throw away all the non-wffs that turn up, leaving us with an effective enumeration of all the wffs. ⊠

Proof sketch for (ii) Proved similarly, except at the last stage we throw away the non-sentences (i.e. throw away the open wffs as well as the mere symbol salads which aren't wffs at all). ⊠

Proof sketch for (iii) Assume that T-proofs are linear sequences of wffs. Just as we can enumerate all the possible wffs, so we can effectively enumerate all the possible sequences of wffs in some 'alphabetical order'. One brute-force way is to start enumerating all possible strings of symbols, and throw away any that isn't a sequence of wffs. By the definition of an axiomatized theory, there is then an algorithmic recipe for deciding which of these sequences of wffs are well-formed proofs in the theory (since for each wff it is decidable whether it is either an axiom or follows from previous wffs in the list by allowed inference moves). So as we go along we can mechanically select out the proof sequences from the other sequences of wffs, to give us an effective enumeration of all the

where '$\vdash$' signifies the relation of formal deducibility, and '$\vDash$' signifies the relation of semantic consequence. As it happens, the first proof of the semantic completeness of a proof-system for quantificational logic was also due to Gödel, and the result is often referred to as 'Gödel's Completeness Theorem' (Gödel, 1929). The topic of *that* theorem is therefore evidently not to be confused with his (First) Incompleteness Theorem, which concerns the negation incompleteness of certain theories of arithmetic.

possible proofs. (If T-proofs are more complex arrays of wffs – as in tree systems – then the construction of an effective enumeration of the arrays needs to be correspondingly more complex: but the core proof-idea remains the same.) ⊠

Proof sketch for (iv) Start effectively enumerating proofs. But this time, just record their conclusions (when those are sentences, i.e. closed wffs, which is decidable if T is an axiomatized formal theory). This mechanically generated list now contains all and only the theorems of the theory. ⊠

Two comments about these proof sketches. First, our talk about listing strings of symbols in 'alphabetical order' can be cashed out in various ways. In fact, *any* systematic mechanical ordering will do here. Here's one simple device (it prefigures the use of 'Gödel numbering', which we'll encounter later). Suppose, to keep things easy, the theory has a basic alphabet of less than ten symbols (this is no real restriction). With each of the basic symbols of the theory we correlate a different digit from '1, 2, . . . , 9'; we will reserve '0' to indicate some punctuation mark, say a comma. So, corresponding to each finite sequence of symbols there will be a sequence of digits, which we can read as expressing a number. For example: suppose we set up a theory using just the symbols

$$\neg,\ \rightarrow,\ \forall,\ (,\ ),\ \mathsf{F},\ \mathsf{x},\ \mathsf{c},\ '$$

and we associate these symbols with the digits '1' to '9' in order. Then e.g. the wff

$$\forall\mathsf{x}(\mathsf{Fx}\ \rightarrow\ \neg\forall\mathsf{x}'\neg\mathsf{F}''\mathsf{cx}')$$

(where 'F''' is a two-place predicate) would be associated with the number

$$374672137916998795$$

We can now list off the wffs constructible from this vocabulary as follows. We examine each number in turn, from 1 upwards. It will be decidable whether the standard base-ten numeral for that number codes a sequence of the symbols which forms a wff, since we are dealing with a formal theory. If the number does correspond to a wff φ, we enter φ onto our list of wffs. In this way, we mechanically produce a list of wffs – which obviously must contain all wffs since to any wff corresponds some numeral by our coding. Similarly, taking each number in turn, it will be decidable whether its numeral corresponds to a series of symbols which forms a sequence of wffs separated by commas (remember, we reserved '0' to encode commas).

Our second comment is this. We should be very clear that to say that the theorems of a formal axiomatized theory can be mechanically *enumerated* is not to say that the theory is *decidable*. It is one thing to have a mechanical method which is bound to generate any theorem eventually; it is quite another thing to have a mechanical method which, given an arbitrary wff φ, can determine – without going on for ever – whether φ will ever turn up on the list of theorems.

3.6 Negation complete theories are decidable

Despite that last point, however, we do have the following important result in the special case of negation-complete theories:[10]

> **Theorem 3** *A consistent, axiomatized, negation-complete formal theory T is decidable.*

Proof We know from Theorem 2 that there's an algorithm for effectively enumerating the theorems of T. So start effectively listing the theorems. Let φ be any sentence of T. Since, by hypothesis, T is negation-complete, either φ is a theorem of T or $\neg\varphi$ is. So it is guaranteed that – within a finite number of steps – either φ or $\neg\varphi$ will be produced in our enumeration of the theorems. If φ is produced, stop the enumeration, and we can conclude that φ is a theorem. If on the other hand $\neg\varphi$ is produced, stop the enumeration, and we can conclude that φ is not a theorem, since the theory is assumed to be consistent. Hence, in this case, there *is* a dumbly mechanical procedure for deciding whether φ is a theorem. ⊠

We are, of course, relying here on our ultra-generous notion of decidability-in-principle we explained above (in Section 2.1). We might have to twiddle our thumbs for an immense time before one of φ or $\neg\varphi$ to turn up. Still, our 'wait and see' method is guaranteed in this case to produce a result in finite time, in an entirely mechanical way – so this counts as an effectively computable procedure in our official generous sense.

[10] By the way, it is trivial that an *inconsistent* axiomatized theory with a classical logic is decidable. For if T is inconsistent, it entails every wff of T's language by the classical principle *ex contradictione quodlibet.* So all we have to do to determine whether φ is a T-theorem is to decide whether φ is a sentence of T's language, which by hypothesis you can if T is an axiomatized formal theory.

4 Capturing numerical properties

The previous two chapters concerned axiomatized formal theories in general. This chapter introduces some key concepts we need in describing formal arithmetics in particular, notably the concepts of *expressing* and *capturing* numerical properties. But we need to start with two quick preliminary sections, about notation and about the very idea of a property.

4.1 Three remarks on notation

(a) Gödel's First Incompleteness Theorem is about the limitations of axiomatized formal theories of arithmetic: if a theory T satisfies some fairly minimal constraints, we can find arithmetical truths that T can't prove. Evidently, in discussing Gödel's result, it will be *very* important to be clear about when we are working 'inside' some specified formal theory T and when we are talking informally 'outside' that particular theory (e.g. in order to establish truths that T can't prove).

However, we do want our informal talk to be compact and perspicuous. Hence we will tend to borrow the standard logical notation from our formal languages for use in augmenting mathematical English (so, for example, we will write '$\forall x \forall y(x + y = y + x)$' as a compact way of expressing the 'ordinary' arithmetic truth that the order in which you sum numbers doesn't matter).

Equally, we will want our formal wffs to be readable. Hence we will tend to use notation in building our formal languages that is already familiar from informal mathematics (so, for example, if we want to express the addition function in a formalized theory of arithmetic, we will use the usual sign '$+$', rather than some unhelpfully anonymous two-place function symbol like 'f^2_3').

This two-way borrowing of notation will inevitably make expressions of informal everyday arithmetic and their formal counterparts look very similar. And while context alone should no doubt make it pretty clear which is which, it is best to have a way of explicitly marking the distinction. To that end, *we will adopt the convention of using a* sans-serif *font for expressions in our formal languages.* Thus compare . . .

$$\begin{array}{cc}
\forall x \forall y(x + y = y + x) & \forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} + \mathsf{y} = \mathsf{y} + \mathsf{x}) \\
\exists y\, y = S0 & \exists \mathsf{y}\, \mathsf{y} = \mathsf{S0} \\
1 + 2 = 3 & \mathsf{1} + \mathsf{2} = \mathsf{3}
\end{array}$$

The expressions on the left will belong to our mathematicians'/logicians' augmented English (borrowing 'S' to mean 'the successor of'): the expressions on the right are wffs – or abbreviations for wffs – of one of our formal languages, with the symbols chosen to be reminiscent of their intended interpretations.

(b) In addition to *italic symbols* for informal mathematics and sans-serif symbols for formal wffs, we also need another layer of symbols. For example, we need a compact way of generalizing about formal expressions, as when we defined negation completeness in Section 3.4 by saying that for any wff φ, the theory T implies either φ or its negation $\neg\varphi$. We'll standardly use Greek letters for this kind of 'metalinguistic' duty. We will also occasionally use Greek letters like 'ξ' and 'ζ' as place-holders, indicating gaps to be filled in expressions. Note then that these symbols again belong to logicians' augmented English: Greek letters will never belong to our formal languages themselves.

So what exactly is going on when we are talking about a formal language $\mathcal{L}$ and say e.g. that the negation of φ is $\neg\varphi$, when we are apparently mixing a symbol from augmented English with a symbol from $\mathcal{L}$? Answer: there are hidden quotation marks, and '$\neg\varphi$' is to be read as meaning 'the expression that consists of the negation sign "$\neg$" followed by φ'.

(c) Sometimes, when being *very* punctilious, logicians use so-called Quine-quotes when writing mixed expressions containing both formal and metalinguistic symbols (thus: $\ulcorner\neg\varphi\urcorner$). But this is excessive. We are not going to bother, and no one will get confused by our more casual (and entirely standard) practice. In any case, we'll want to use corner-quotes later for a different purpose.

We'll be very relaxed about ordinary quotation marks too. We've so far been rather punctilious about using them when mentioning, as opposed to using, wffs and other formal expressions. But from now on, we will normally drop them other than around single symbols. Again, no confusion should ensue.

Finally, we will also be pretty relaxed about dropping unnecessary brackets in formal expressions (and we'll change the shape of pairs of brackets, and occasionally insert redundant ones, when that aids readability).

4.2 A remark about extensionality

The extension of the numerical property P is the set of numbers n such that n is P. And here's a stipulation: *we are going to use 'property' talk in this book in such a way that P and Q count as the same property if they have the same extension.* As the jargon has it, we are treating properties extensionally. (There's nothing unusual about this stipulation in logical contexts: we are just being explicit about our practice to fend off possible misunderstandings.)

Now, just as one and the same thing can be picked out by two denoting terms, so a property can be presented in different ways. The number two is picked out by both the terms 'the smallest prime' and 'the cube root of eight': as philosophers are apt to put it, although these terms have different senses, they have the same reference. Likewise, the numerical predicates '. . . divides by two' and '. . . is the predecessor of an odd number' also have different senses, but locate the same property. For a more dramatic example, if Goldbach's conjecture is true, '…is even and greater than two' locates the same property as '…is even and the sum of two primes'. But very evidently, the two phrases have quite different senses (and no-one knows if they really *do* have the same extension).

4.3 The language L_A

Now to business. There is no single language which could reasonably be called *the* language for formal arithmetic: rather, there is quite a variety of different languages, apt for framing theories of different strengths.

However, the core theories of arithmetic which we'll be discussing first are mostly framed in the language L_A, i.e. the interpreted language $\langle \mathcal{L}_A, \mathcal{I}_A \rangle$, which is a formalized version of what we called 'the language of basic arithmetic' in Section 1.2. So let's begin by characterizing this language.

(a) *Syntax* The logical vocabulary of $\mathcal{L}_A$ comprises the usual connectives and brackets, an inexhaustible supply of variables (including, let's suppose, 'a' to 'z'), the usual first-order quantifiers, plus the identity symbol. The fine details are not critical.

The non-logical vocabulary of $\mathcal{L}_A$ is $\{\mathsf{0}, \mathsf{S}, +, \times\}$, where

'$\mathsf{0}$' is a constant.

'S' is a one-place function-expression (read 'the successor of').

'$+$' and '$\times$' are two-place function-expressions.

For readability, we'll allow ourselves to write e.g. $(\mathsf{x} + \mathsf{y})$ and $(\mathsf{x} \times \mathsf{y})$ rather than $+(\mathsf{x}, \mathsf{y})$ and $\times(\mathsf{x}, \mathsf{y})$.

We'll now define the (standard) *numerals* and the *terms* of $\mathcal{L}_A$. Numerals, then, are expressions that you can build up from our single constant '$\mathsf{0}$' using just the successor function, i.e. they are expressions of the form $\mathsf{SS}\ldots\mathsf{S0}$ with zero or more occurrences of 'S'. We'll abbreviate the numerals $\mathsf{S0}$, $\mathsf{SS0}$, $\mathsf{SSS0}$, etc. by $\mathsf{1}$, $\mathsf{2}$, $\mathsf{3}$, etc. And when we want to generalize, we'll write 'n' to indicate the standard numeral $\mathsf{SS}\ldots\mathsf{S0}$ with n occurrences of 'S'.[1]

[1] In using 'S' rather than 's', we depart from the normal logical practice which we follow elsewhere of using upper-case letters for predicates and lower-case letters for functions: but this particular departure is sanctioned by aesthetics and common usage.

We've made two other notational choices here. First, a very common alternative convention is to use a postfixed prime as the symbol for the successor function; in that notation the standard numerals are then $\mathsf{0}$, $\mathsf{0}'$, $\mathsf{0}''$, $\mathsf{0}'''$, …. (Although we won't be using that notation in this book, I'll avoid using the prime symbol for other purposes when there could be any possibility of a browsing reader mistaking it for an unintended successor symbol.)

Second, another very common convention is to an *overlined* symbol like $\overline{n}$ to abbreviate the numeral $\mathsf{SS}\ldots\mathsf{S0}$ with n occurrences of 'S'. Our alternative convention of using a sans serif font for formal expressions *and* abbreviations for formal expressions avoids the visual clutter of overlinings.

Next, terms are expressions that you can build up from '$\mathsf{0}$' and/or variables using the successor function S, addition and multiplication – as in $\mathsf{SSS0}$, $(\mathsf{S0} + \mathsf{x})$, $(\mathsf{SSS0} \times (\mathsf{Sx} + \mathsf{y}))$, and so on. Putting it more carefully,

> '$\mathsf{0}$' is a term, as is any variable.
>
> If σ and τ are terms, so are $\mathsf{S}\sigma$, $(\sigma + \tau)$, $(\sigma \times \tau)$.
>
> Nothing else is a term.

The *closed* terms are the variable-free terms.

Now, the only predicate built into $\mathcal{L}_A$ is the identity sign. So that means that the only possible atomic wffs have the form $\sigma = \tau$, where again σ and τ are terms. Then wffs are formed from atomic wffs in the entirely standard way, by using connectives and quantifiers.

(b) *Semantics* The interpretation $\mathcal{I}_A$ gives items of $\mathcal{L}_A$'s non-logical vocabulary their natural readings. In particular, $\mathcal{I}_A$ assigns values to closed terms as follows:

> The value of '$\mathsf{0}$' is zero. Or in an obvious shorthand, $val[\,\mathsf{0}\,] = 0$.
>
> If τ is a closed term, then $val[\mathsf{S}\tau] = val[\tau] + 1$.
>
> If σ and τ are closed terms, then $val[(\sigma + \tau)] = val[\sigma] + val[\tau]$, and $val[(\sigma \times \tau)] = val[\sigma] \times val[\tau]$.

It immediately follows, by the way, that numerals have the values that they should have, i.e. for all n, $val[\mathsf{n}] = n$.

The atomic sentences (closed wffs) of $\mathcal{L}_A$ must all have the form $\sigma = \tau$, where σ and τ are closed terms. Like any relational sentence, such a sentence is true if the pair of values of the featured constants satisfies the featured predicate. Hence

> A sentence of the form $\sigma = \tau$ is true iff $val[\sigma] = val[\tau]$.

Molecular sentences built up using the truth-functional connectives are then evaluated in the obvious ways: so

> A sentence of the form $\neg\varphi$ is true iff φ is not true.
>
> A sentence of the form $(\varphi \wedge \psi)$ is true iff φ and ψ are both true.

and so on through the other connectives. Which leaves us the quantified sentences to deal with. Following the line in Section 3.2, we could explicitly say that the domain of quantification is the natural numbers, and a sentence of the form $\exists \mathsf{v}\varphi(\mathsf{v})$ is true on $\mathcal{I}_A$ just if there is some number in the domain which we can dub with a new constant 'c' so that – on a suitable expansion of the interpretation $\mathcal{I}_A$ – $\varphi(\mathsf{c})$ comes out true. But of course, each number in the intended domain *already* has a term to pick it out, i.e. the numeral n. So, here – in this special case – we can drop the explicit talk of a domain of quantification and new constants and put the rule for the existential quantifier very simply like this:

> A sentence of the form $\exists\nu\varphi(\nu)$ (where 'ν' can be any variable) is true iff, for some number n, $\varphi(\mathsf{n})$ is true.

Similarly

> A sentence of the form $\forall\nu\varphi(\nu)$ is true iff, for any n, $\varphi(\mathsf{n})$ is true.

And then it is easy to see that $\mathcal{I}_A$ will, as we want, effectively assign a unique truth-condition to every $\mathcal{L}_A$ sentence.

(c) *Just one comment* The semantics $\mathcal{I}_A$ straightforwardly entails that the sentence $(1 + 2) = 3$, i.e. in unabbreviated form $(\mathsf{S0} + \mathsf{S00}) = \mathsf{SSS0}$, is true just so long as one plus two is three; likewise $\exists\mathsf{x}\ 4 = (\mathsf{x} \times 2)$, i.e. $\exists\mathsf{x}\ \mathsf{S0000} = (\mathsf{x} \times \mathsf{S00})$, is true just so long as there is some number such that four is twice that number (i.e. so long as four is even). But, by any normal arithmetical standards, one plus two *is* three, and four *is* even. So by the same workaday standards, those two L_A-sentences are indeed true.

Later, when we come to present Gödel's Theorems, we'll describe how to construct some much more complex sentences of an arithmetical theory T built in the language L_A, sentences which are 'true but unprovable-in-T'. But while the sentences in question are exotic, there is nothing in the least exotic about the notion of truth being applied to them here: it is the very same workaday notion we've just so simply explained. $\mathcal{I}_A$ explicitly defines what it takes for any L_A-sentence, however complex, to be true in this humdrum sense.[2]

Now there are, to be sure, philosophers who will say that no L_A-sentence *is* strictly speaking true in the humdrum sense – because they are equally prepared to say that, strictly speaking, one plus two *isn't* three and four *isn't* even.[3] Such common-or-garden arithmetic claims, they aver, presuppose the existence of numbers as mysterious kinds of objects in some Platonic heaven, and they doubt the literal existence of such things. On their view, then, such arithmetical entities are fictions: and, at least when we are on our very best behaviour, we really ought to say only that *in the arithmetical fiction*, one plus two equals three, and four is even. We can't, however, tangle with this rather popular view here: and fortunately we needn't do so, for the issues it raises are quite orthogonal to

[2] The notion of truth, in other words, is not being used here in a sense carrying any special weight. Wittgenstein puts a related point this way in lectures:

> One often hears statements about 'true' and 'false' – for example, that there are true mathematical statements which can't be proved in *Principia Mathematica*, etc. In such cases the thing is to avoid the words 'true' and 'false' altogether, and to get clear that to say that p is true is just to assert p; and to say that p is false is simply to deny p or to assert $\neg p$. It is not a question of whether p is 'true in a different sense'. It is a question of whether we assert p. (Wittgenstein, 1989, p. 188)

Here Wittgenstein is talking of the use of 'true' applied *within* a language, while we are currently concerned about the use *across* languages, when we talk about L_A in English. The point becomes: claiming an L_A sentence is true commits us to nothing more than an assertion of a corresponding English arithmetical sentence.

[3] See e.g. (Field, 1989, Ch. 1) and (Balaguer, 1998) for discussion.

our main concerns in this book. Fictionalists about arithmetic can systematically read our talk of various L_A being true in their favoured way – i.e. as talk 'within the arithmetical fiction'.

4.4 Expressing numerical properties and relations

A competent formal theory of arithmetic should surely be able to talk about a lot more than just the successor function, addition and multiplication. But 'talk about' *how*?

(a) Let's assume for the moment that we are dealing with a theory built in the language L_A. So, for a first example, consider L_A-sentences of the type

1. $\psi(\mathsf{n}) =_{\text{def}} \exists\mathsf{v}(2 \times \mathsf{v} = \mathsf{n})$

Then, for example, if $n = 4$, '$\psi(\mathsf{n})$' unpacks into '$\exists\mathsf{v}(\mathsf{SS0} \times \mathsf{v} = \mathsf{SSSS0})$'. It is obvious that, for any n,

if n is even, then $\psi(\mathsf{n})$ is true,
if n isn't even, then $\neg\psi(\mathsf{n})$ is true,

where we mean, of course, true on the arithmetic interpretation built into L_A. Relatedly, then, consider the open wff with one free variable

1′. $\psi(\mathsf{x}) =_{\text{def}} \exists\mathsf{v}(2 \times \mathsf{v} = \mathsf{x})$

This is, as the logicians say, satisfied by the number n just when $\psi(\mathsf{n})$ is true, i.e. just when n is even – or to put it another way, $\psi(\mathsf{x})$ has the set of even numbers as its extension. Which means that our open wff expresses the property *even*, for it has the right extension.

Another example: n has the property of being prime if it is greater than one, and its only factors are one and itself. Or equivalently, n is prime just in case it is not 1, and of any two numbers that multiply to give n, one of them must be 1. So consider the wff

2. $\chi(\mathsf{n}) =_{\text{def}} (\mathsf{n} \neq 1 \;\wedge\; \forall\mathsf{u}\forall\mathsf{v}(\mathsf{u} \times \mathsf{v} = \mathsf{n} \;\rightarrow\; (\mathsf{u} = 1 \;\vee\; \mathsf{v} = 1)))$

(where we use $\alpha \neq \beta$ to abbreviate $\neg\alpha = \beta$). This holds just in case n is prime, i.e. for every n,

if n is prime, then $\chi(\mathsf{n})$ is true,
if n isn't prime, then $\neg\chi(\mathsf{n})$ is true.

So relatedly, the corresponding open wff

2′. $\chi(\mathsf{x}) =_{\text{def}} (\mathsf{x} \neq 1 \;\wedge\; \forall\mathsf{u}\forall\mathsf{v}(\mathsf{u} \times \mathsf{v} = \mathsf{x} \;\rightarrow\; (\mathsf{u} = 1 \;\vee\; \mathsf{v} = 1)))$

is satisfied by exactly the prime numbers. Hence $\chi(\mathsf{x})$ expresses the property *prime*, again in the sense of having the right extension.

In this sort of way, a formal language like L_A with limited basic resources can come to express a whole variety of arithmetical properties by means of complex open wffs with the right extensions. And our examples motivate the following official definition that applies to *any* language L in which we can form the standard numerals:

> A property P is *expressed* by the open wff $\varphi(\mathsf{x})$ with one free variable in an arithmetical language L iff, for every n,
>
> if n has the property P, then $\varphi(\mathsf{n})$ is true,
>
> if n does not have the property P, then $\neg\varphi(\mathsf{n})$ is true.

'True' of course continues to mean true on the given interpretation built into the relevant L.

(b) We can now extend our definition in the obvious way to cover relations. Note, for example, that in a language like L_A

3. $\psi(\mathsf{m}, \mathsf{n}) =_{\mathrm{def}} \exists\mathsf{v}(\mathsf{v} + \mathsf{m} = \mathsf{n})$

is true just in case $m \leq n$. And so it is natural to say that the corresponding expression

3′. $\psi(\mathsf{x}, \mathsf{y}) =_{\mathrm{def}} \exists\mathsf{v}(\mathsf{v} + \mathsf{x} = \mathsf{y})$

expresses the relation *less-than-or-equal-to*, in the sense of getting the extension right. Generalizing again:

> A two-place relation R is expressed by the open wff $\varphi(\mathsf{x}, \mathsf{y})$ with two free variables in an arithmetical language L iff, for any m, n,
>
> if m has the relation R to n, then $\varphi(\mathsf{m}, \mathsf{n})$ is true,
>
> if m does not have the relation R to n, then $\neg\varphi(\mathsf{m}, \mathsf{n})$ is true.

Likewise for many-place relations.[4]

[4] A footnote for very-well-brought-up logicians. We could have taken the canonical way of expressing a monadic property to be not a complete open wff $\varphi(\mathsf{x})$ but a *predicative* expression $\varphi(\xi)$ – where 'ξ' here isn't a variable but a *place-holder*, marking a *gap* to be filled by a term (i.e. by a name or variable). Similarly, we could have taken the canonical way of expressing a two-place relation to be a doubly gappy predicative expression $\varphi(\xi, \zeta)$, etc. Now, there are pernickety philosophical reasons for preferring the gappy notation to express properties and relations. However, it is the default informal mathematical practice to prefer to use complete expressions with free variables rather than expressions with place-holders to mark the gaps; and sticking to this practice makes for a more familiar-looking notation and so greatly aids readability. (Trust me! – I did at one stage try writing this book systematically using the notation with Greek letters as place-holders and some passages looked quite unnecessarily rebarbative.)

Still, there's a wrinkle. Just once, in Section 10.5, we'll want to talk about the expressive power of a theory whose language lacks quantifiers and variables, so in particular lacks expressions with free variables. In that special context, you'll have to treat any reference to expressions of the form $\varphi(\mathsf{x}, \mathsf{y})$ as a cheerful abuse of notation, with the apparent variables really functioning as place-holders, so there we mean what we really should otherwise write as $\varphi(\xi, \zeta)$ and so on.

4.5 Capturing numerical properties and relations

Of course, we don't merely want various properties of numbers to be *expressible* in the language of a formal theory of arithmetic. We also want to be able to use the theory to *prove* facts about which numbers have which properties.

Now, it is a banal observation that to establish facts about *individual* numbers typically requires much less sophisticated proof-techniques than proving general truths about *all* numbers. To take a dramatic example, there's a school-room mechanical routine for testing any given even number to see whether it is the sum of two primes. But while, case by case, every even number other than two that has ever been checked passes the test, no one knows how to prove Goldbach's conjecture – i.e. no one knows how to prove 'in one fell swoop' that *every* even number greater than two is the sum of two primes.

Let's focus then on the relatively unambitious task of case-by-case proving that particular numbers have or lack a certain property. This level of task is reflected in the following definition concerning formal provability:

> The theory T *captures* the property P by the open wff $\varphi(\mathsf{x})$ iff, for any n,
>
> if n has the property P, then $T \vdash \varphi(\mathsf{n})$,
> if n does not have the property P, then $T \vdash \neg\varphi(\mathsf{n})$.

For example, in theories of arithmetic T with very modest axioms, the wff $\psi(\mathsf{x}) =_{\text{def}} \exists\mathsf{v}(\mathsf{2} \times \mathsf{v} = \mathsf{x})$ not only expresses but captures the property *even.* In other words, for each even n, T can prove $\psi(\mathsf{n})$, and for each odd n, T can prove $\neg\psi(\mathsf{n})$. Likewise, in the same theories, the wff $\chi(\mathsf{x})$ from the previous section not only expresses but captures the property *prime.*

As you would expect, extending the notion of 'capturing' to the case of relations is straightforward:

> The theory T *captures* the two-place relation R by the open wff $\varphi(\mathsf{x}, \mathsf{y})$ iff, for any m, n,
>
> if m has the relation R to n, then $T \vdash \varphi(\mathsf{m}, \mathsf{n})$
> if m does not have the relation R to n, then $T \vdash \neg\varphi(\mathsf{m}, \mathsf{n})$.

4.6 Expressing vs. capturing: keeping the distinction clear

A little later, we'll need the notion of a formal theory's capturing numerical *functions* (as well as properties and relations). But there is a minor complication in that case, so let's not delay over it here; instead we'll immediately press on in the next chapter to apply the concepts that we've already defined.

However, I should pause to note frankly that my talk of a theory's 'capturing' a numerical property is a bit deviant. But terminology here varies anyway. Perhaps most commonly these days, logicians talk of P being 'represented' by a $\varphi(\mathsf{x})$ satisfying our conditions for capture. But I'm unapologetic: experience suggests that the more usual jargon is in some danger of engendering confusion in beginners. While 'capture' is helpfully mnemonic for 'case-by-case prove'.

But whatever your favoured jargon, the key thing is to be absolutely clear about the distinction we need to mark – so let's highlight it again. Whether a property P is *expressible* in a given theory just depends on the richness of that theory's *language*. Whether a property P can be *captured* by the theory depends on the richness of its *proof-system*.[5]

Expressibility does not imply capturability: indeed, we will prove later that – for any respectable theory of arithmetic T – there are numerical properties that are expressible but not capturable in T (see e.g. Section 15.6). However, there is a link in the other direction. Suppose T is a *sound* theory of arithmetic – i.e. one whose axioms are true on the given arithmetic interpretation of its language and whose logic is truth-preserving. Then T's theorems are all true. Hence if $T \vdash \varphi(\mathsf{n})$, then $\varphi(\mathsf{n})$ is true. And if $T \vdash \neg\varphi(\mathsf{n})$, then $\neg\varphi(\mathsf{n})$ is true. Which shows that *if $\varphi(\mathsf{x})$ captures P in the sound theory T then, a fortiori, T's language expresses P.*

[5] 'Expresses' is used in our way by e.g. Smullyan (1992, p. 19). As alternatives, we find e.g. 'arithmetically defines' (Boolos et al., 2002, p. 199), or simply 'defines' (Leary, 2000, p. 130), (Enderton, 2002, p. 205).

Gödel originally talked of a numerical relation being 'decidable' (*entscheidungsdefinit*) when it is captured by an arithmetical wff (Gödel, 1931, p. 176). As later alternatives to our 'captures' we find 'numeralwise expresses' (Kleene, 1952, p. 195), (Fisher, 1982, p. 112), and also simply 'expresses'(!) again (Mendelson, 1997, p. 170), 'formally defines' (Tourlakis, 2003, p. 180) and plain 'defines' (Boolos et al., 2002, p. 207). At least 'binumerate' – (Smorynski, 1977, p. 838), (Lindström, 2003, p. 9) – won't cause confusion. But as noted, 'represents' (although it is perhaps too close for comfort to 'expresses') seems the most common choice in recent texts: see e.g. (Leary, 2000, p. 129), (Enderton, 2002, p. 205), (Cooper, 2004, p. 56).

The moral is plain: when reading other discussions, always very carefully check the local definitions of the jargon!

5 Sufficiently strong arithmetics

In Chapter 6, we'll begin examining some formal theories of arithmetic 'from the bottom up', in the sense of first setting down the axioms of the theories and then exploring what the various theories are capable of proving. Here in this chapter, however, we proceed the other way about. We introduce the concept of a *sufficiently strong* theory of arithmetic, which is a theory that by definition *can* prove what we'd *like* any moderately competent theory of arithmetic to be able to prove about decidable properties of particular numbers. We then establish some easy but quite deep results about such theories.

5.1 The idea of a 'sufficiently strong' theory

Suppose that P is some effectively decidable property of numbers, i.e. one for which we have a mechanical algorithm for deciding, given a natural number n, whether n has property P or not (see Section 2.1).

Now, when we construct a formal theory of the arithmetic of the natural numbers, we will surely want deductions inside our theory to be able to track, case by case, any mechanical calculation that we can already perform informally. After all, we don't want going formal to *diminish* our ability to determine whether n has this property P. As we stressed in Section 3.1, formalization aims at regimenting what we can already do: it isn't supposed to hobble our efforts. So while we might have some passing interest in more limited theories, we will mainly want to aim for a formal theory T which at least (a) is able to frame some open wff $\varphi(\mathsf{x})$ which expresses the decidable property P, and (b) is such that if n has property P, $T \vdash \varphi(\mathsf{n})$, and if n does not have property P, $T \vdash \neg\varphi(\mathsf{n})$ – i.e. we want T to capture P (in the sense of Section 4.5).

The suggestion therefore is that, if P is any decidable property of numbers, we ideally want a competent theory of arithmetic T to be able to capture P. Which motivates the following definition:

> A formal theory of arithmetic T is *sufficiently strong* if it captures all decidable numerical properties.

So it seems a reasonable and desirable condition on a formal theory of the arithmetic of the natural numbers that it be sufficiently strong. Much later (in Section **??**), when we've done some close analysis of the general idea of effective decidability, we'll finally be in a position to warrant the claim that some simple and (by then) very familiar theories do indeed meet this condition, and we'll thereby show that the condition of being 'sufficiently strong' is actually easily met. But we can't establish that now: this chapter just supposes that there *are* such theories and derives some consequences.[1]

5.2 An undecidability theorem

A trivial way for a theory T to be sufficiently strong (i.e. to prove lots of wffs about properties of individual numbers) is by being inconsistent (i.e. by proving *every* wff about individual numbers). It goes without saying, however, that we are interested in *consistent* theories.

We also like to get *decidable* theories when we can, i.e. theories for which there is an algorithm for determining whether a given wff is a theorem (see Section 3.4). But, sadly, we have the following key result:[2]

> **Theorem 4** *No consistent, sufficiently strong, axiomatized formal theory of arithmetic is decidable.*

Proof We suppose T is a consistent and sufficiently strong axiomatized theory yet also decidable, and derive a contradiction.

By hypothesis, T's language can frame open wffs with 'x' free. These will be effectively enumerable: $\varphi_0(\mathsf{x}), \varphi_1(\mathsf{x}), \varphi_2(\mathsf{x}), \ldots$ For by Theorem 2 (Section 3.5), we know that the complete set of sentences of T can be effectively enumerated. It will then be a mechanical business to select out the ones with just 'x' free (there are standard rules for determining whether a variable is free or bound).

And now let's fix on the following definition:

n has the property D if and only if $T \vdash \neg\varphi_n(\mathsf{n})$

Note that the construction here links the subscripted index with the standard numeral substituted for the variable in $\neg\varphi_n(\mathsf{x})$. So this is a cousin of the 'diagonal' construction which we encountered in Section 2.2 (see the comment (c) on the proof of Theorem 1).

We next show that the supposition that T is a decidable theory entails that the 'diagonal' property D is an effectively decidable property of numbers. For given any number n, it will be a mechanical matter to enumerate the open wffs until the n-th one, $\varphi_n(\mathsf{x})$, is produced. Then it is a mechanical matter to form the numeral n, substitute it for the variable and prefix a negation sign. Now we just apply the supposed mechanical procedure for deciding whether a sentence

[1] It is in fact rather more usual to define being 'sufficiently strong' as a matter of capturing not only all decidable properties but also all decidable relations and all computable functions too. But since we haven't yet defined what it is to capture a function, and since the arguments of this chapter in any case don't depend on that notion, we might as well stick with our weaker definition of sufficient strength.

[2] The undecidability of arithmetic was first proved in (Church, 1936). The direct proof given here can be extracted from Theorem 1 of (Tarski et al., 1953, pp. 46–49). The first published version of our informal version which I know is (Hunter, 1971, pp. 224–225), though T.J. Smiley was presenting it in Cambridge lectures in the 1960s.

is a T-theorem to test whether the wff $\neg\varphi_n(\mathsf{n})$ is a theorem. So, on our current assumptions, there is an algorithm for deciding whether n has the property D.

Since, by hypothesis, the theory T is sufficiently strong, it can capture all decidable numerical properties: so it follows that, in particular, D is capturable by some open wff. This wff must of course occur somewhere in our enumeration of the $\varphi(\mathsf{x})$. Let's suppose the d-th wff does the trick: that is to say, property D is captured by $\varphi_d(\mathsf{x})$.

It is now entirely routine to get out a contradiction. For, by definition, to say that $\varphi_d(\mathsf{x})$ captures D means that for any n,

> if n has the property D, $T \vdash \varphi_d(\mathsf{n})$,
> if n doesn't have the property D, $T \vdash \neg\varphi_d(\mathsf{n})$.

So taking in particular the case $n = d$, we have

i. if d has the property D, $T \vdash \varphi_d(\mathsf{d})$,
ii. if d doesn't have the property D, $T \vdash \neg\varphi_d(\mathsf{d})$.

But note that our initial definition of the property D implies in particular:

iii. d has the property D if and only if $T \vdash \neg\varphi_d(\mathsf{d})$.

From (ii) and (iii), it follows that whether d has property D or not, the wff $\neg\varphi_d(\mathsf{d})$ is a theorem either way. So by (iii) again, d does have property D, hence by (i) the wff $\varphi_d(\mathsf{d})$ must be a theorem too. So a wff and its negation are both theorems of T. Therefore T is inconsistent, contradicting our initial assumption that T is consistent.

In sum, the supposition that T is a consistent and sufficiently strong axiomatized formal theory of arithmetic *and* decidable leads to contradiction. ⊠

Which is a beautiful result: indeed it is one of the delights of our topic: we can get exciting theorems fast!

There's an old hope (which goes back to Leibniz) that can be put in modern terms like this: we might one day be able to mechanize mathematical reasoning to the point that a suitably primed computer could solve all mathematical problems in a domain by deciding theoremhood in an appropriate formal theory. What we've just shown is that this is a false hope: as soon as a theory is strong enough to capture all boringly mechanical reasoning about individual numbers, it must cease to be decidable.

5.3 An incompleteness theorem

Now let's put together Theorem 3 (established in Section 3.6) and Theorem 4.

> **Theorem 3** *A consistent, axiomatized, negation-complete formal theory is decidable.*
>
> **Theorem 4** *No consistent, sufficiently strong, axiomatized formal theory of arithmetic is decidable.*

These, of course, immediately entail

> **Theorem 5** *A consistent, sufficiently strong, axiomatized formal theory of arithmetic cannot also be negation complete.*

That is to say, for any c.s.s.a. (consistent, sufficiently strong, axiomatized) theory of arithmetic, there will be a pair of sentences φ and $\neg\varphi$, neither of which are theorems. But one of these must be true on the given interpretation of T's language. Therefore, for any c.s.s.a. theory of arithmetic T, there are true-but-unprovable wffs in T.

And adding in new axioms won't help. To re-play the sort of argument we gave in Section 1.2, suppose T is a c.s.s.a. theory of arithmetic, and suppose G_T is a true sentence of arithmetic that T can't prove or disprove. The theory U which you get by adding G_T as a new axiom to T will, of course, now trivially prove G_T, so we've plugged that gap. But note that U is consistent (for if U, i.e. $T + \mathsf{G}_T$, were inconsistent, then by reductio, $T \vdash \neg\mathsf{G}_T$, contrary to hypothesis). And U is sufficiently strong (since it can still prove everything T can prove). It is still decidable which wffs are axioms of U, so the theory still counts as a properly axiomatized formal theory. So Theorem 5 applies, and the new c.s.s.a. theory U must therefore contain a wff G_U (distinct from G_T, of course) which is again true-on-interpretation but unprovable. So T is not only *incomplete* but in a good sense *incompletable.*

5.4 The truths of arithmetic can't be axiomatized

Here's another pair of definitions.

i. A set of wffs Σ is *axiomatizable* if there is an axiomatized formal theory T such that, for any wff φ, $\varphi \in \Sigma$ if and only if $T \vdash \varphi$ (i.e. Σ is the set of T-theorems).

ii. An interpreted language L is *sufficiently rich* if it can express every decidable property of numbers.

Then, as an immediate corollary of Theorem 5, we have

> **Theorem 6** *The set of truths of a sufficiently rich language L is unaxiomatizable.*

Proof Suppose that L is sufficiently rich, and we'll suppose – for reductio – that the set of true wffs of L *can* be axiomatized by a theory T. Then T must be negation complete – since for every closed wff ψ of L, either ψ or $\neg\psi$ is true, and by hypothesis the true one is a theorem.

But let P be any decidable property of numbers. Since L is sufficiently rich, there is some $\varphi(\mathsf{x})$ such that, for any n,

if n has the property P, $\varphi(\mathsf{n})$ is true,
if n doesn't have the property P, $\neg\varphi(\mathsf{n})$ is true.

Since T entails all the truths, it follows that for any n

> if n has the property P, $T \vdash \varphi(\mathsf{n})$,
> if n doesn't have the property P, $T \vdash \neg\varphi(\mathsf{n})$.

Since P was an arbitrary decidable property, this means that T must be sufficiently strong (by definition of the notion of sufficient strength). But T is consistent, since by hypothesis it only contains truths. So, by Theorem 5, T is not negation-complete after all. Contradiction! ⊠

Now, the informal idea of (all) 'the truths of arithmetic' is no doubt not a sharp one. But however we refine it, presumably we want it to include at least the truths about the nice, decidable, properties of numbers. So in our jargon, the truths of arithmetic, on any plausible sharpening of that idea, should be the truths of a sufficiently rich language. Hence our new theorem warrants the informal claim expressed in the title of this section: the truths of arithmetic can't be axiomatized.

Interlude: taking stock, looking ahead

Theorem 5, our informal incompleteness theorem, isn't the same as Gödel's First Incompleteness Theorem. But it is a cousin, and it looks to be a quite terrific result to arrive at so very quickly.

Or is it? Everything depends, for a start, on the idea of a 'sufficiently strong' theory of arithmetic which captures *all* decidable properties of numbers. Now, as we've already briefly indicated in Section 2.1, there are a number of standard, well-understood, fully coherent ways of formally refining the intuitive notion of decidability, ways that turn out to locate the same entirely definite and well-defined class of numerical properties (in fact, these are the properties whose application can be decided by a Turing machine). The specification 'all decidable properties of numbers' is therefore in good order. And hence so is the idea of a theory being strong enough to capture all decidable properties of numbers.

But that doesn't take us very far. For it could still be the case that a theory that captures all decidable properties has to be very rich indeed – involving (say) a language with an infinity of different fundamental predicates for the infinity of different decidable properties, with each new predicate waiting to be governed by its own special axioms. So couldn't the moral of our Theorem just be that there can't be complete theories of all the arithmetical truths expressible in certain ultra-rich languages? That would still leave open the possibility that there could be complete theories governing the propositions expressible in some more restricted languages like L_A, the language of basic arithmetic.

However, we announced right back in Section 1.2 that Gödel's own result rules out complete theories even of the truths of basic arithmetic. Hence, if our easy Theorem 5 is to have the full reach of Gödel's Theorem, we'll need to show that *a theory with the restricted language of basic arithmetic* can already be sufficiently strong.

The state of play is therefore this: if our informal style of argument for Theorem 5 is to be used to establish something like Gödel's own result, then it needs to be augmented with (i) a general treatment of the class of decidable properties, *and* (ii) a proof that some axiomatized theory of basic arithmetic can indeed capture all such properties.

But even with (i) and (ii) in play, there would still remain a very significant difference between our easy Theorem and Gödel's theorem. For our easy result would still only tell us that *somewhere or other* there's an unprovable truth in any sufficiently strong T. By contrast, Gödel's proof of the official First Incompleteness Theorem actually tells us how to take a theory T and construct a true but unprovable-in-T sentence (the one that encodes 'I am unprovable in T').

Moreover, Gödel shows that this unprovable-in-T sentence has a particularly simple form: it is a wff of the kind $\forall \mathsf{y}\varphi(\mathsf{y})$, where each separate instance $\varphi(\mathsf{n})$ *is* provable-in-T (so the unprovable wff is, as it were, only just out of reach). And Gödel shows all this – albeit still after quite an amount of hard work – without needing the general treatment in (i) and without needing all of (ii) either.

In sum, then, there *is* a very significant gap between our intriguing, quickly-derived, but informal Theorem 5 and the industrial-strength First Incompleteness Theorem that Gödel proves. So, while what we have shown so far is highly suggestive, it is time to start turning to Gödel's own arguments.

To avoid getting lost, it will help to keep in mind the following road-map of the route we are taking, leading up to the Theorem:

1. We begin by describing some standard formal systems of arithmetic, in particular the benchmark system PA, so-called 'First-order Peano Arithmetic', and an important subsystem Q, 'Robinson Arithmetic'. (Chapters 6–8)

2. These systems are framed in L_A, the language of basic arithmetic. So they only have the successor, addition and multiplication as 'built-in' functions. But we go on to describe the large family of 'primitive recursive' functions, properties and relations (which includes all familiar arithmetical functions like the factorial and exponential, and familiar arithmetic properties like being prime, and relations like one number being the square of another). And we then show that Q and PA can capture all the primitive recursive functions, properties and relations – a result that was, in essence, first proved by Gödel. (Chapters 9–11)

3. We next turn to Gödel's simple but crucial innovation – the idea of systematically associating expressions of a formal arithmetic with numerical codes. Any sensibly systematic scheme of 'Gödel numbering' will do: but Gödel's original style of numbering has a certain naturalness, and makes it tolerably straightforward to prove arithmetical results about the codings. With a coding scheme in place, we can reflect properties and relations of strings of symbols of PA (to concentrate on that theory) by properties and relations of their Gödel numbers. For example, we can define the numerical properties *Term* and *Wff* which hold of a number when it is the code number for a symbol sequence which is, respectively, a term or a wff of PA. And we can, crucially, define the numerical relation $Prfseq(m, n)$ which holds when m codes for an array of wffs that is a PA proof, and n codes the closed wff that is thereby proved. This project of coding up various syntactic relationships is often referred to as *the arithmetization of syntax*. And what Gödel showed next is that – given a sane system of Gödel numbering – these and a large family of related arithmetical properties and relations are primitive recursive. (The outline idea here

is beautifully simple: joining up the dots takes some tiresome work in Chapter 12.)

4. Next – the really exciting bit! – we use the fact that relations like *Prfseq* are expressible in PA to construct the 'Gödel sentence' G. Given the coding scheme, G 'says' there is no number that is the Gödel number of a PA proof of the wff that results from a certain construction – where the wff that results is none other than G itself. So in effect G 'says' of itself 'I am unprovable in PA'. We can then show that G is indeed unprovable, assuming PA is consistent. So we've found an arithmetical wff that is true but unprovable in PA. (And given a slightly stronger assumption than PA's consistency, ¬G must also be unprovable in PA.) Moreover, this unprovable wff has the simple form we indicated above. (Chapter 13)

5. Finally, Gödel notes that the true-but-unprovable sentence G for PA is generated by a method that can be applied to any other arithmetic that satisfies some modest conditions. In particular, adding G as a new axiom to PA just gives us a revised theory for which we can generate another true-but-unprovable wff G'. PA is therefore not only incomplete but incompletable. Indeed, *any* properly axiomatized that contains the weaker theory Q is incompletable. (Chapter 14)

Now, on the face of it, all that looks rather different from our informal proof using the idea of 'sufficiently strong' theories which capture all computable properties. But a close connection can be made by noting that the primitive recursive properties are in fact a sub-class of the intuitively computable properties. Hence showing that Q and PA can capture all primitive recursive properties is at least a precursor to showing those theories are sufficiently strong. However, only when we have travelled the original Gödelian route (which doesn't presuppose a *general* account of computability) will we return to formalize the argument of our informal proof (a task which *does* presuppose such a general account).

6 Two formalized arithmetics

We now move on from the generalities of the previous chapters, and look at four particular formal theories of arithmetic. We limber up with two simple theories in this chapter, namely Baby Arithmetic and Robinson Arithmetic. And then in Chapter 8 we introduce Peano Arithmetic (the strongest and most important of our four theories) and also briefly look at Presburger Arithmetic (a cut-down version of Peano Arithmetic). These theories differ in strength, but they do share the following features:

1. Zero and functions like successor and addition are treated as primitive notions governed by basic axioms, and are not defined in terms of anything more fundamental.
2. The theories' deductive apparatus is no stronger than familiar first-order logic. So we can quantify over *numbers*, but there are no second-order quantifiers, so we can't quantify over *numerical properties*.

It is absolutely standard to start by considering formal theories of arithmetic with these features, though later in this book we'll briefly look at some theories which lack them.

6.1 BA – Baby Arithmetic

We begin with a *very* simple formal arithmetic which 'knows' about the addition of particular numbers, 'knows' its multiplication tables, but can't express general facts about numbers at all (it lacks the whole apparatus of quantification). Hence our label *baby arithmetic*, or BA for short. As with any formal theory, we need to characterize its language, deductive apparatus, and axioms:

(a) BA's language is $L_B = \langle \mathcal{L}_B, \mathcal{I}_B \rangle$. $\mathcal{L}_B$'s non-logical vocabulary is the same as that of $\mathcal{L}_A$ (Section 4.3): so there is a single individual constant '0', the one-place function symbol 'S', and the two-place function symbols '+' and '×'. Note that $\mathcal{L}_B$ in particular contains the standard numerals. However, $\mathcal{L}_B$'s logical apparatus is restricted. As we said, it lacks the quantifiers and variables. But it has the identity sign (so that we can express equalities), and negation (so that we can express inequalities): and we might as well give it the other propositional connectives too.

The intended interpretation $\mathcal{I}_B$ is the obvious one. '0' still has the value zero. 'S' signifies the successor function, and '+' and '×' are interpreted as addition and multiplication.

(b) BA's deductive apparatus can be based on your favourite system of propositional logic to deal with connectives. Then we need to add some standard rules to deal with the identity sign: in particular, we need a version of Leibniz's Law. Then, if τ and ρ are closed terms (see Section 4.3, (a)), Leibniz's Law will allow us to infer $\varphi(\rho)$ from the premisses $\varphi(\tau)$ and $\tau = \rho$ or $\rho = \tau$.

(c) Now for the axioms of BA. To start with, we want to pin down at least the following facts about the structure of the number sequence: (1) Zero is the *first* number, i.e. isn't a successor; so for every n, $0 \neq Sn$. (2) The number sequence never circles back on itself; so different numbers have different successors – or contraposing, for any m, n, if $Sm = Sn$ then $m = n$.

We haven't got quantifiers in BA's language, however, so we can't express these general facts directly. Rather, we need to employ *schemas* and say: *any sentence that you get from one of the following schemas by substituting standard numerals for the place-holders 'ζ', 'ξ' is an axiom.*

Schema 1 $\quad \mathsf{0} \neq \mathsf{S}\zeta$

Schema 2 $\quad \mathsf{S}\zeta = \mathsf{S}\xi \rightarrow \zeta = \xi$

We'll quickly show that instances of these schemas do indeed entail that different terms in the sequence $\mathsf{0}$, $\mathsf{S0}$, $\mathsf{SS0}$, $\mathsf{SSS0}$, ..., pick out different numbers. Recall, we use 'n' to represent the numeral $\mathsf{SS}\ldots\mathsf{S0}$ with n occurrences of 'S': so the result we need is that, for any m, n, if $m \neq n$, then $\mathsf{BA} \vdash \mathsf{m} \neq \mathsf{n}$.

Proof Suppose $m \neq n$ (and let $|m-n|-1 = j \geq 0$). And assume $\mathsf{m} = \mathsf{n}$ as a temporary supposition in BA (that's a supposition of the form $\mathsf{SS}\ldots\mathsf{S0} = \mathsf{SS}\ldots\mathsf{S0}$, with m occurrences of 'S' on the left and n on the right). We can now use instances of Schema 2 plus modus ponens to repeatedly strip off initial occurrences of 'S', one on each side of the identity, until either (i) we derive $\mathsf{0} = \mathsf{Sj}$, or else (ii) we derive $\mathsf{Sj} = \mathsf{0}$ and then use the symmetry of identity to conclude $\mathsf{0} = \mathsf{Sj}$. But $\mathsf{0} \neq \mathsf{Sj}$ is an axiom (an instance of Schema 1). Contradiction. So, $\mathsf{m} \neq \mathsf{n}$ follows by reductio. Which proves that if $m \neq n$, then $\mathsf{BA} \vdash \mathsf{m} \neq \mathsf{n}$. ⊠

Next we pin down the addition function by saying that any wff that you get by substituting numerals in the following is also an axiom:

Schema 3 $\quad \zeta + \mathsf{0} = \zeta$

Schema 4 $\quad \zeta + \mathsf{S}\xi = \mathsf{S}(\zeta + \xi)$

Instances of Schema 3 tell us the result of adding zero. Instances of Schema 4 with 'ξ' replaced by '$\mathsf{0}$' tell us how to add one (i.e. add $S0$) in terms of adding zero and then applying the successor function to the result. Once we know about adding one, we can use another instance of Schema 4 with 'ξ' replaced by '$\mathsf{S0}$' to tell us how to add two ($SS0$) in terms of adding $S0$. We can then invoke the same Schema again to tell us how to add three ($SSS0$) in terms of adding two: and so on and so forth, thus defining addition for every natural number.

We can similarly pin down the multiplication function by requiring every numeral instance of the following to be axioms too:

Schema 5 $\quad \zeta \times 0 = 0$

Schema 6 $\quad \zeta \times \mathsf{S}\xi = (\zeta \times \xi) + \zeta$

Instances of Schema 5 tell us the result of multiplying by zero. Instances of Schema 6 with 'ξ' replaced by '0' tell us how to multiply by one in terms of multiplying by zero and then applying the already-defined addition function. Once we know about multiplying by one, we can use another instance of Schema 4 with 'ξ' replaced by 'S0' to tell us how to multiply by two in terms of adding one. And so on and so forth, thus defining multiplication for every number.

Note, it is evidently decidable whether a wff is an instance of one of the six Schemas, and so it is decidable whether a wff is an axiom of BA, as is required if BA is to count as an axiomatized theory.

6.2 BA is complete

It is easy to see that BA's axioms can be used to prove all the correct results about the addition or multiplication of two numbers.

To illustrate, here's a BA derivation of $2 \times 1 = 2$, or rather (putting that in unabbreviated form) of $\mathsf{SS0} \times \mathsf{S0} = \mathsf{SS0}$.

1.	$\mathsf{SS0} \times 0 = 0$	Instance of Schema 5
2.	$\mathsf{SS0} \times \mathsf{S0} = (\mathsf{SS0} \times 0) + \mathsf{SS0}$	Instance of Schema 6
3.	$\mathsf{SS0} \times \mathsf{S0} = 0 + \mathsf{SS0}$	From 1, 2 by LL

('LL' of course indicates the use of Leibniz's Law which allows us to intersubstitute identicals.) To proceed, we now need to show that $0 + \mathsf{SS0} = \mathsf{SS0}$ – and note, this *isn't* an instance of Schema 3. So

4.	$0 + 0 = 0$	Instance of Schema 3
5.	$0 + \mathsf{S0} = \mathsf{S}(0 + 0)$	Instance of Schema 4
6.	$0 + \mathsf{S0} = \mathsf{S0}$	From 4, 5 by LL
7.	$0 + \mathsf{SS0} = \mathsf{S}(0 + \mathsf{S0})$	Instance of Schema 4
8.	$0 + \mathsf{SS0} = \mathsf{SS0}$	From 6, 7 by LL

Which gives us what we want:

9.	$\mathsf{SS0} \times \mathsf{S0} = \mathsf{SS0}$	From 3, 8 by LL

That's pretty laborious, but it works. And a moment's reflection on this little proof reveals that similar proofs will enable us to derive the value of *any* sum or product of two numerals.

Let's say that an *equation* of BA is a wff of the form $\tau = \rho$, where τ and ρ are closed terms. Then we have the following pair of general results about equations:

1. If $\tau = \rho$ is true, then $\mathsf{BA} \vdash \tau = \rho$.
2. If $\tau = \rho$ is false, then $\mathsf{BA} \vdash \tau \neq \rho$.

In other words, in the jargon of Section 3.4, BA correctly decides all equations.

Proof sketch for (1) Our sample proof above illustrates the sort of BA derivation that will prove any true simple equation of the type $\mathsf{j} + \mathsf{k} = \mathsf{m}$ or $\mathsf{j} \times \mathsf{k} = \mathsf{n}$. Given a more complex closed term τ, involving nested additions and multiplications (or applications of the successor function), we can then prove a true wff of the form $\tau = \mathsf{t}$ with a numeral on the right by repeated steps of evaluating inner-most brackets.

To take a mini-example, suppose τ has the shape $\mathsf{SS}((\mathsf{j} + \mathsf{k}) + \mathsf{S}(\mathsf{j} \times \mathsf{k}))$. Then we first prove the identities $\mathsf{j} + \mathsf{k} = \mathsf{m}$ and $\mathsf{j} \times \mathsf{k} = \mathsf{n}$, evaluating the inner-most bracketed expressions. Substituting these results into the logical truth $\tau = \tau$ using Leibniz's Law will enable us to derive

$$\mathsf{SS}((\mathsf{j} + \mathsf{k}) + \mathsf{S}(\mathsf{j} \times \mathsf{k})) = \mathsf{SS}(\mathsf{m} + \mathsf{Sn})$$

Now evaluate the new, simpler, bracketed expression on the right by proving something of the form $\mathsf{m} + \mathsf{Sn} = \mathsf{t}$. Hence, using Leibniz's Law again, we get

$$\mathsf{SS}((\mathsf{j} + \mathsf{k}) + \mathsf{S}(\mathsf{j} \times \mathsf{k})) = \mathsf{SSt}$$

And we are done, as the expression on the right is a numeral. Evidently, this method of repeated substitutions always works: for *any* complex closed term τ we'll be able to prove a wff correctly equating its value to that of some numeral.

So, generalizing further, given any *two* closed terms τ and ρ, if they have the same value so $\tau = \rho$ is true, then we'll be able to prove $\tau = \rho$ by proving each term equal to the same numeral. ⊠

Proof sketch for (2) Suppose that two complex closed terms τ and ρ have values m and n, where $m \neq n$. By the argument for (1), we'll then be able to prove a pair of wffs of the form $\tau = \mathsf{m}$, $\rho = \mathsf{n}$. But we've already shown in the previous section that if $m \neq n$, BA proves $\mathsf{m} \neq \mathsf{n}$. So, if $m \neq n$, a BA proof of $\tau \neq \rho$ follows using Leibniz's Law twice. ⊠

These two results in turn imply

Theorem 7 BA *is negation complete.*

Proof sketch Note that $\mathcal{L}_B$, like $\mathcal{L}_A$, has only one primitive predicate, the identity relation. So the only atomic claims expressible in BA are equations involving closed terms; all other sentences are truth-functional combinations of such equations. But we've just seen that we can (1) prove each true 'atom' and (2) prove the negation of each false 'atom'. So, by a theorem of propositional logic, we can derive any true truth-functional combination of atoms (equations), i.e. prove any true sentence. And we can derive the negation of false truth-functional combination of atoms (equations), i.e. disprove any false sentence. In short, in the jargon of Section 3.4, BA correctly decides every sentence. Hence, for any sentence φ of BA, since either φ or $\neg\varphi$ is true, either φ or $\neg\varphi$ is a theorem. So BA is negation-complete. ⊠

Since BA is complete, it is decidable, by Theorem 3. But of course we don't need a brute-force search through possible derivations in order to determine whether a sentence φ is a BA theorem. For note that all BA theorems are true (since the axioms are); and all true BA-sentences are theorems (as we've just seen). Hence determining whether the BA-sentence φ is true settles whether it is a theorem. But any such φ expresses a truth-function of equations, so we can mechanically work out whether it is true or not by using school-room arithmetic for the equations and then using a truth-table.

6.3 Q – Robinson Arithmetic

So far, then, so straightforward. But the reason that Baby Arithmetic manages to prove every correct claim *that it can express* – and is therefore negation complete by our definition – is that it can't express very much. In particular, it can't express any generalizations at all. BA's completeness comes at the high price of being expressively extremely impoverished.

The obvious way to start beefing up BA into something more exciting is to restore the familiar apparatus of quantifiers and variables. So let's keep the same non-logical vocabulary, but now allow ourselves the full resources of first-order logic, so that we are working with the full language $L_A = \langle \mathcal{L}_A, \mathcal{I}_A \rangle$ of basic arithmetic (see Section 4.3). Q's deductive apparatus can be your favourite version of first-order logic with identity, whatever that is.

Since we now have the quantifiers available to express generality, we can replace each metalinguistic Schema (specifying an infinite number of particular axioms) by a single object-language Axiom. For example, we can replace the first two Schemas governing the successor function by

Axiom 1 $\forall \mathsf{x}(0 \neq \mathsf{Sx})$

Axiom 2 $\forall \mathsf{x}\forall \mathsf{y}(\mathsf{Sx} = \mathsf{Sy} \to \mathsf{x} = \mathsf{y})$

Each instance of our earlier Schemas 1 and 2 can be deduced from the corresponding Axiom by one or two applications of Universal Instantiation.

Note, however, that while these Axioms tell us that zero isn't a successor, they leave it open that there are other objects that aren't successors cluttering up the domain of quantification (there could be 'pseudo-zeros'). We don't want our quantifiers – now that we've introduced them – running over such stray objects: so let's now explicitly rule them out:

Axiom 3 $\forall \mathsf{x}(\mathsf{x} \neq 0 \ \to\ \exists \mathsf{y}(\mathsf{x} = \mathsf{Sy}))$

Next, we can similarly replace our previous Schemas for addition and multiplication by universally quantified Axioms:

Axiom 4 $\forall \mathsf{x}(\mathsf{x}+0=\mathsf{x})$

Axiom 5 $\forall \mathsf{x}\forall \mathsf{y}(\mathsf{x}+\mathsf{Sy}=\mathsf{S}(\mathsf{x}+\mathsf{y}))$

Axiom 6 $\forall \mathsf{x}(\mathsf{x}\times 0=0)$

Axiom 7 $\forall \mathsf{x}\forall \mathsf{y}(\mathsf{x}\times \mathsf{Sy}=(\mathsf{x}\times \mathsf{y})+\mathsf{x})$

The formalized theory with language L_A, Axioms 1 to 7, plus a standard first-order logic, is called *Robinson Arithmetic*, or (very often) simply Q.[1]

6.4 Q is not complete

Since any BA Axiom – i.e. any instance of one of our previous Schemas – can be derived from one of our new Q Axioms, every $\mathcal{L}_B$-sentence that can be proved in BA is equally a quantifier-free $\mathcal{L}_A$-sentence which can be proved in Q. Hence, Q again correctly decides every quantifier-free sentence.

However, there are very simple true quantified sentences that Q can't prove. For example, Q can prove any particular wff of the form $0+\mathsf{n}=\mathsf{n}$. But it can't prove the universal generalization $\chi =_{\text{def}} \forall \mathsf{x}(0+\mathsf{x}=\mathsf{x})$.

Proof sketch One standard strategy for showing that a wff χ is *not* a theorem of a given theory T is to find an interpretation (often a deviant, unintended interpretation) for the T-wffs which makes the axioms of T true and hence all its theorems true, but which makes χ false.

Employing this strategy, we want to find a deviant, unintended, interpretation of Q's Axioms which would make them true but for which 'adding' a 'number' to zero changes it. Here's an artificial – but still legitimate – example.

Take the domain of our deviant, unintended, interpretation of Q to be the set N^* comprising the natural numbers but with two other 'rogue' elements a and b added (these can be Gwyneth Paltrow and Chris Martin, or any other pair that takes your fancy). Let '0' still to refer to zero. And take 'S' now to pick out the successor* function S^* which is defined as follows: $S^*n = Sn$ for any natural number in the domain, while for our rogue elements $S^*a = a$, and $S^*b = b$. It is immediate that Axioms 1 to 3 are still true on this deviant interpretation.

We now need to extend this model to re-interpret Q's function '+'. Suppose we take this to pick out addition*, where $m +^* n = m + n$ for any natural numbers m, n in the domain, while $a +^* n = a$ and $b +^* n = b$. Further, for any x (whether number or rogue element), $x +^* a = b$ and $x +^* b = a$. It is easily checked that interpreting '+' as addition* still makes Axioms 4 and 5 true. But by construction, $0 +^* a \neq a$, so this interpretation indeed makes χ false.

We are not quite done, however, as we still need to show that we can give a co-ordinate re-interpretation of '×' in Q by some deviant multiplication* function. But we can leave it as an exercise to fill in suitable details. ⊠

[1] This formal system was first isolated in (Robinson, 1952) and immediately became well-known through the classic (Tarski et al., 1953).

So Q can't prove χ. But obviously, Q can't prove $\neg\chi$ either. Just revert to the standard interpretation $\mathcal{I}_A$. Q certainly has true axioms on this interpretation: so all theorems are true on $\mathcal{I}_A$. But $\neg\chi$ is false on $\mathcal{I}_A$, so it can't be a theorem.

In sum, $\mathsf{Q} \nvdash \chi$ and $\mathsf{Q} \nvdash \neg\chi$.[2] Which gives us the utterly unsurprising

Theorem 8 Q *is not negation complete.*

Of course, we've already announced that Gödel's First Theorem is going to prove that *no* axiomatized theory in the language of basic arithmetic can be negation-complete. But what we've just shown is that we don't need to invoke anything as Gödel's arguments to see that Q is incomplete: Q is, so to speak, *boringly* incomplete.

6.5 Why Q is interesting

Given it can't even prove $\forall \mathsf{x}(0 + \mathsf{x} = \mathsf{x})$, Q is a pretty weak theory of arithmetic. Even so, despite such shortcomings, Q will emerge as having some very nice properties which do make it of special interest. In particular, and perhaps rather surprisingly, it turns out to be *sufficiently strong* in the sense of Chapter 5. For 'sufficient strength' is a matter of being able to case-by-case prove enough wffs about decidable properties of individual numbers. And Q's weakness at proving generalizations doesn't stop it doing that.

Establishing this claim is business for much later (plainly, we can only establish it when we have a general theory of decidability to hand). But it gives us the motivation to press on in the next chapter to look in a preliminary way at what Q *can* prove.

[2] The notational shorthand here is to be read in the obvious way: i.e we write $T \nvdash \varphi$ as short for not-($T \vdash \varphi$), i.e. φ is unprovable in T.

7 What Q can prove

In this chapter, we explore something of what Q can establish. Unavoidably, the detailed arguments get a bit fiddly; and so you are welcome to skip the proofs of stated *results* (you won't miss anything of conceptual interest). But you will need to carry forward to later chapters an understanding of some key *concepts* we'll be introducing: so do at least skim through.

Here's a quick guide through the sections of this chapter.

1. We first show that the wff $\exists \mathsf{v}(\mathsf{v} + \mathsf{x} = \mathsf{y})$ not only expresses but captures the relation *less-than-or-equal-to* in Q (cf. Section 4.4, (b)). This motivates our adding the symbol '$\leq$' to Q so that $\xi \leq \zeta$ is a definitional abbreviation of $\exists \mathsf{v}(\mathsf{v} + \xi = \zeta)$.

2. We then show that Q can formally prove a range of expected results about the less-than-or-equals-to relation.

3. Consider a *bounded* universal quantification, as in e.g. $(\forall \mathsf{x} \leq \mathsf{n})\varphi(\mathsf{x})$ when read in the obvious way. That quantification is equivalent to the finite conjunction $\varphi(\mathsf{0}) \wedge \varphi(\mathsf{1}) \wedge \ldots \wedge \varphi(\mathsf{n})$. Likewise a *bounded* existential quantification is equivalent to a finite disjunction. 'Simple' L_A sentences (or in the official jargon, Δ_0 sentences) that are built up with bounded quantifiers are therefore like the quantifier-free sentences which they are equivalent to: it is a simple matter of mechanical calculation to determine whether they are true or not.

4. We show that Q knows enough about bounded quantifiers to be able to correctly decide every 'simple' sentence – i.e. prove it if it is true, disprove it if it is false. And Q can also prove any true existentially quantified 'simple' wff (in the official jargon, can prove any true Σ_1 sentence).

5. We finally note an intriguing corollary of that last result.

7.1 Capturing *less-than-or-equal-to* in Q

In this section, we'll show that the *less-than-or-equal-to* relation is captured by the wff $\exists \mathsf{v}(\mathsf{v} + \mathsf{x} = \mathsf{y})$ in Q. That is to say, for any particular pair of numbers, Q can prove that the first is less than or equal to the second (if it is) or prove that it isn't (if it isn't).

Proof sketch Suppose $m \leq n$, so for some $k \geq 0$, $k + m = n$. Q can prove everything BA proves and hence, in particular, can prove every true equation. So we have $\mathsf{Q} \vdash \mathsf{k} + \mathsf{m} = \mathsf{n}$. But $\mathsf{k} + \mathsf{m} = \mathsf{n} \vdash \exists \mathsf{v}(\mathsf{v} + \mathsf{m} = \mathsf{n})$ by existential quantifier introduction. Therefore $\mathsf{Q} \vdash \exists \mathsf{v}(\mathsf{v} + \mathsf{m} = \mathsf{n})$, as was to be shown.

Suppose alternatively $m > n$. We need to show $\mathsf{Q} \vdash \neg\exists \mathsf{v}(\mathsf{v} + \mathsf{m} = \mathsf{n})$. We'll first demonstrate this in the case where $m = 2$, $n = 1$. Then it will be be easy to see that the style of proof will work more generally.

Now, we were very free-and-easy about the style of proof-system we gave to Q. However, if we are to give illustrative proofs, we've got to plump for one definite proof-system or another. Because it is well-known (but also simple to follow if you don't know it), we will work with a standard Fitch-style natural deduction system. This has two features. First, we use symbols to act as temporary names ('parameters'): you can think of these as recruited from our stock of unused variables. Second, we indent sub-proofs while a new temporary assumption is in force.

So consider the following laborious argument with some inference steps slightly compressed (LL of course indicates the use of Leibniz's Law):

1.	$\exists \mathsf{v}(\mathsf{v} + \mathsf{SS0} = \mathsf{S0})$	Supposition
2.	$\mathsf{a} + \mathsf{SS0} = \mathsf{S0}$	Supposition
3.	$\mathsf{a} + \mathsf{SS0} = \mathsf{S}(\mathsf{a} + \mathsf{S0})$	From Axiom 5
4.	$\mathsf{S}(\mathsf{a} + \mathsf{S0}) = \mathsf{S0}$	From 2, 3 by LL
5.	$(\mathsf{a} + \mathsf{S0}) = \mathsf{S}(\mathsf{a} + 0)$	From Axiom 5
6.	$\mathsf{SS}(\mathsf{a} + 0) = \mathsf{S0}$	From 4, 5 by LL
7.	$(\mathsf{a} + 0) = \mathsf{a}$	From Axiom 4
8.	$\mathsf{SSa} = \mathsf{S0}$	From 6, 7 by LL
9.	$\mathsf{SSa} = \mathsf{S0} \to \mathsf{Sa} = 0$	From Axiom 2
10.	$\mathsf{Sa} = 0$	From 8, 9 by MP
11.	$0 = \mathsf{Sa}$	From 10
12.	$0 \neq \mathsf{Sa}$	From Axiom 1
13.	Contradiction!	From 11, 12
14.	Contradiction!	∃E 1, 2–13
15.	$\neg\exists \mathsf{v}(\mathsf{v} + \mathsf{SS0} = \mathsf{S0})$	RAA 1–14.

The only step to explain is at line (14) where we use a version of Existential Elimination: the idea is that if the supposition $\varphi(\mathsf{a})$ leads to contradiction, for arbitrary a, then $\exists \mathsf{v}\varphi(\mathsf{v})$ also leads to contradiction. Inspection of our proof immediately reveals that we can indeed use the same basic pattern of argument to show that $\mathsf{Q} \vdash \neg\exists \mathsf{v}(\mathsf{v} + \mathsf{m} = \mathsf{n})$ whenever $m > n$. So we are done. $\boxtimes$

Given that result, it is evidently appropriate now to add the standard symbol '$\leq$' to L_A, defined so that (whatever we put for 'ξ' and 'ζ'), $\xi \leq \zeta =_{\mathrm{def}} \exists \mathsf{v}(\mathsf{v} + \xi = \zeta)$.[1] Since it greatly helps readability, we'll henceforth make very free use of this symbol.

[1] Actually, we really need to be a bit more careful than that in stating the rule for unpacking the abbreviation, if we are to avoid any possible clash of variables. But we're not going to fuss about the details.

And while we are at it, let's adopt two further abbreviatory conventions. We will very often want to express *bounded* quantifications, i.e. to say that all or some numbers *less than or equal to a given number* satisfy some particular condition. Hence we'll be making frequent use of wffs of the form $\forall\xi(\xi \leq \kappa \to \varphi)$ and $\exists\xi(\xi \leq \kappa \,\wedge\, \varphi)$. It is standard to abbreviate such wffs by $(\forall\xi \leq \kappa)\varphi$ and $(\exists\xi \leq \kappa)\varphi$ respectively.

7.2 Eight simple facts about what Q can prove

We could now go on to prove, example by example, that various other properties and relations can be captured in Q. However, it just isn't worth treating more examples in a piecemeal way because – as we said at the end of the previous chapter – we are later going to be able to show that Q can capture *all* decidable properties and relations. In fact Q is custom designed to be about the weakest tidy theory that has this property.

In this section, then, we'll note some basic general facts about what Q can prove, chosen with an eye to what we need for the later proof that Q is indeed 'sufficiently strong'. Here's eight:

1. For any n, $\mathsf{Q} \vdash \forall \mathsf{x}(\mathsf{Sx} + \mathsf{n} = \mathsf{x} + \mathsf{Sn})$.
2. For any n, $\mathsf{Q} \vdash \forall \mathsf{x}(\{\mathsf{x} = 0 \vee \mathsf{x} = 1 \vee \ldots \vee \mathsf{x} = \mathsf{n}\} \to \mathsf{x} \leq \mathsf{n})$.
3. For any n, $\mathsf{Q} \vdash \forall \mathsf{x}(\mathsf{x} \leq \mathsf{n} \to \{\mathsf{x} = 0 \vee \mathsf{x} = 1 \vee \ldots \vee \mathsf{x} = \mathsf{n}\})$.
4. For any n, if $\mathsf{Q} \vdash \varphi(0)$, $\mathsf{Q} \vdash \varphi(1)$, $\ldots$, $\mathsf{Q} \vdash \varphi(\mathsf{n})$,
   then $\mathsf{Q} \vdash (\forall \mathsf{x} \leq \mathsf{n})\varphi(\mathsf{x})$.
5. For any n, if $\mathsf{Q} \vdash \varphi(0)$, or $\mathsf{Q} \vdash \varphi(1)$, $\ldots$, or $\mathsf{Q} \vdash \varphi(\mathsf{n})$,
   then $\mathsf{Q} \vdash (\exists \mathsf{x} \leq \mathsf{n})\varphi(\mathsf{x})$.
6. For any n, $\mathsf{Q}\ \forall \mathsf{x}(\mathsf{x} \leq \mathsf{n} \ \to\ \mathsf{x} \leq \mathsf{n} + 1)$.
7. For any n, $\mathsf{Q} \vdash \forall \mathsf{x}(\mathsf{n} \leq \mathsf{x} \ \to\ (\mathsf{n} = \mathsf{x} \ \vee\ \mathsf{Sn} \leq \mathsf{x}))$.
8. For any n, $\mathsf{Q} \vdash \forall \mathsf{x}(\mathsf{x} \leq \mathsf{n} \ \vee\ \mathsf{n} \leq \mathsf{x})$.

Some of these facts are just a bit tiresome to demonstrate: but none of the proofs is very exciting or involves anything deep. If you have a taste for logical brainteasers, then see how many of the results you can establish before reading on. If you don't, just make sure that you understand the content of these results;

We should remark, by the way, that some presentations treat '≤' as a primitive symbol built into our formal theories from the start, governed by its own additional axiom(s). Nothing important hangs on the difference between that approach and our policy of introducing the symbol by definition. And nothing hangs either on our policy of introducing '≤' rather than '<', which can be defined by $\xi < \zeta =_{\mathrm{def}} \exists \mathsf{v}(\mathsf{Sv} + \xi = \zeta)$.

and then feel free to skim the following proofs very lightly, or even skip them altogether.[2]

Proof for (1) The following holds for arbitrary a:

$$\mathsf{Sa} + \mathsf{n} \;=\; \mathsf{Sa} + \overbrace{\mathsf{SS}\ldots\mathsf{S}}^{n \text{ times}}\mathsf{0} \;=\; \overbrace{\mathsf{SS}\ldots\mathsf{S}}^{n+1 \text{ times}}\;\mathsf{a} \;=\; \mathsf{a} + \mathsf{S}\overbrace{\mathsf{SS}\ldots\mathsf{S}}^{n \text{ times}}\mathsf{0} \;=\; \mathsf{a} + \mathsf{Sn}$$

Each of the middle two equations is proved by the repeated use of Axiom 5 (to shift around, one by one, the occurrences of 'S' after the plus signs) and then one appeal to Axiom 4. Since a was arbitrary, we can universally generalize to get the desired result. ⊠

Proof for (2) Suppose we are given $\mathsf{a} = \mathsf{k}$, for some $k \leq n$. Then

Suppose $\mathsf{a} = \mathsf{k}$ (where $k \leq n$). Then, since Q proves all true equations, it proves $(\mathsf{n} - \mathsf{k}) + \mathsf{a} = \mathsf{n}$, and hence $\exists\mathsf{v}(\mathsf{v} + \mathsf{a} = \mathsf{n})$, i.e. Q proves $\mathsf{a} \leq \mathsf{n}$. So, now suppose $\mathsf{a} = \mathsf{0} \vee \mathsf{a} = \mathsf{1} \vee \ldots \vee \mathsf{a} = \mathsf{n}$. We've just shown each disjunct entails $\mathsf{a} \leq \mathsf{n}$. Hence, arguing by cases, Q proves $\mathsf{a} = \mathsf{0} \vee \mathsf{a} = \mathsf{1} \vee \ldots \vee \mathsf{a} = \mathsf{n} \;\rightarrow\; \mathsf{a} \leq \mathsf{n}$, and then the result is immediate. ⊠

Proof for (3) This is trickier: we'll argue by an informal induction. That's to say, we'll show that (a) the given wff is provable for $n = 0$. Then we show that (b) if it is provable for $n = k$ it is provable for $n = k + 1$. We can conclude that it is therefore provable for all n.[3]

(a) To show the wff is provable for $n = 0$, we need a proof of $\forall\mathsf{x}(\mathsf{x} \leq \mathsf{0} \rightarrow \mathsf{x} = \mathsf{0})$. It is enough to suppose, inside a Q proof, that $\mathsf{a} \leq \mathsf{0}$, for arbitrary a, and deduce $\mathsf{a} = \mathsf{0}$. So suppose $\mathsf{a} \leq \mathsf{0}$, i.e. $\exists\mathsf{v}(\mathsf{v} + \mathsf{a} = \mathsf{0})$. Then for some b, $\mathsf{b} + \mathsf{a} = \mathsf{0}$. Now by Axiom 3, either (i) $\mathsf{a} = \mathsf{0}$, or (ii) $\mathsf{a} = \mathsf{Sa}^\circ$ for some a°. But (ii) implies $\mathsf{b} + \mathsf{Sa}^\circ = \mathsf{0}$, so by Axiom 5 $\mathsf{S}(\mathsf{b} + \mathsf{a}^\circ) = \mathsf{0}$, contradicting Axiom 1. That rules out case (ii). Therefore $\mathsf{a} = \mathsf{0}$ as we needed to show.

(b) We assume that Q proves $\forall\mathsf{x}(\mathsf{x} \leq \mathsf{k} \rightarrow \{\mathsf{x} = \mathsf{0} \vee \mathsf{x} = \mathsf{1} \vee \ldots \vee \mathsf{x} = \mathsf{k}\})$. We want to prove that Q proves $\forall\mathsf{x}(\mathsf{x} \leq \mathsf{k} + \mathsf{1} \rightarrow \{\mathsf{x} = \mathsf{0} \vee \mathsf{x} = \mathsf{1} \vee \ldots \vee \mathsf{x} = \mathsf{k} + \mathsf{1}\})$. It is enough to suppose, inside a Q proof, that $\mathsf{a} \leq \mathsf{k} + \mathsf{1}$, for arbitrary a, and then deduce $\mathsf{a} = \mathsf{0} \vee \mathsf{a} = \mathsf{1} \vee \ldots \vee \mathsf{a} = \mathsf{k} + \mathsf{1}$.

Our supposition, unpacked, is $\exists\mathsf{v}(\mathsf{v} + \mathsf{a} = \mathsf{k} + \mathsf{1})$. By Axiom 3, (i) $\mathsf{a} = \mathsf{0}$ or (ii) $\mathsf{a} = \mathsf{Sa}^\circ$, for some a°. In case (i), the desired long disjunction follows immediately. While in case (ii), we have $\exists\mathsf{v}(\mathsf{v} + \mathsf{Sa}^\circ = \mathsf{Sk})$: hence, by Axiom 5 and Axiom 2, we can derive $\exists\mathsf{v}(\mathsf{v} + \mathsf{a}^\circ = \mathsf{k})$; i.e. $\mathsf{a}^\circ \leq \mathsf{k}$. So, using our given assumption, $\mathsf{a}^\circ = \mathsf{0} \vee \mathsf{a}^\circ = \mathsf{1} \vee \ldots \vee \mathsf{a}^\circ = \mathsf{k}$. Since $\mathsf{a} = \mathsf{Sa}^\circ$, $\mathsf{a} = \mathsf{1} \vee \mathsf{a} = \mathsf{2} \vee \ldots \vee \mathsf{a} = \mathsf{k} + \mathsf{1}$. So putting (i) and (ii) together, $\mathsf{a} = \mathsf{0} \vee \mathsf{a} = \mathsf{1} \vee \mathsf{a} = \mathsf{2} \vee \ldots \vee \mathsf{a} = \mathsf{k} + \mathsf{1}$. ⊠

[2] 'He has only half learned the art of reading who has not added to it the more refined art of skipping and skimming.' (A. J. Balfour, one-time British Prime Minister.) This remark applies in spades to reading mathematical texts!

[3] You need to be very clear what is going on here. We are not using an inductive argument *inside* Q; we can't because Q lacks induction axioms (compare PA in the next chapter, which *does* have induction axioms). Rather, we are standing outside Q and using an everyday informal – or 'metalogical' – reasoning to show something *about* what Q can prove.

Proof for (4) Assume Q proves each of $\varphi(\mathsf{0})$, $\varphi(\mathsf{1})$, $\ldots$, $\varphi(\mathsf{n})$. Then, suppose $\mathsf{a} \leq \mathsf{n}$, for arbitrary a. By Result 3, we have $\mathsf{a} = \mathsf{0} \vee \mathsf{a} = \mathsf{1} \vee \ldots \vee \mathsf{a} = \mathsf{n}$. Now we argue by cases: each disjunct separately – combined with one of our initial suppositions – gives $\varphi(\mathsf{a})$; so we can conclude $\varphi(\mathsf{a})$. Discharging our temporary supposition, $\mathsf{a} \leq \mathsf{n} \to \varphi(\mathsf{a})$. Generalize, and we are done. ☒

Proof for (5) Assume Q proves $\varphi(\mathsf{k})$ for some $k \leq n$. Since Q captures *less-than-than-or-equal-to*, we have $\mathsf{k} \leq \mathsf{n} \ \wedge \ \varphi(\mathsf{k})$, hence $\exists \mathsf{x}(\mathsf{x} \leq \mathsf{n} \ \wedge \ \varphi(\mathsf{x}))$. ☒

Proof for (6) Suppose $\mathsf{a} \leq \mathsf{n}$ for arbitrary a. Then $\mathsf{a} = \mathsf{0} \vee \mathsf{a} = \mathsf{1} \vee \ldots \vee \mathsf{n}$ by Result 3. So by trivial logic, $\mathsf{a} = \mathsf{0} \vee \mathsf{a} = \mathsf{1} \vee \ldots \vee \mathsf{a} = \mathsf{n} \vee \mathsf{a} = \mathsf{n+1}$. So by Result 2, $\mathsf{a} \leq \mathsf{n+1}$. Which gives us what we want. ☒

Proof for (7) Suppose $\mathsf{n} \leq \mathsf{a}$ for arbitrary a. So for some b, $\mathsf{b} + \mathsf{n} = \mathsf{a}$. By Axiom 3, either (i) $\mathsf{b} = \mathsf{0}$ or (ii) $\mathsf{b} = \mathsf{Sb}^{\circ}$ for some b°. If (i), then $\mathsf{0} + \mathsf{SS}\ldots\mathsf{S0} = \mathsf{a}$, and applying Axiom 5 n times and then Axiom 4 gives us $\mathsf{n} = \mathsf{a}$. If (ii), $\mathsf{Sb}^{\circ} + \mathsf{n} = \mathsf{a}$, so by Result 1 we have $\mathsf{b}^{\circ} + \mathsf{Sn} = \mathsf{a}$, hence $\exists \mathsf{v}(\mathsf{v} + \mathsf{Sn} = \mathsf{a})$, i.e. $\mathsf{Sn} \leq \mathsf{a}$. Therefore either way we get $\mathsf{n} = \mathsf{a} \vee \mathsf{Sn} \leq \mathsf{a}$, and we are done. ☒

Proof for (8) We show that (a) the given wff is provable for $n = 0$, and that (b) if it is provable for $n = k$ it is also provable for $n = k + 1$. We can then conclude by another informal induction that the wff is provable for all n.

(a) Note that for any a, $\mathsf{a} + \mathsf{0} = \mathsf{a}$. Hence, $\exists \mathsf{v}(\mathsf{v} + \mathsf{0} = \mathsf{a})$, i.e. $\mathsf{0} \leq \mathsf{a}$. So a fortiori, $\mathsf{0} \leq \mathsf{a} \ \vee \ \mathsf{a} \leq \mathsf{0}$. Generalizing gives us the desired result for $n = 0$.

(b) We'll suppose that the result holds for $n = k$, and show it holds for $n = k + 1$. Hence, for arbitrary a,

i.	$\mathsf{a} \leq \mathsf{k} \ \vee \ \mathsf{k} \leq \mathsf{a}$	By our supposition
ii.	$\mathsf{a} \leq \mathsf{k} \to \mathsf{a} \leq \mathsf{k+1}$	By Result 6
iii.	$\mathsf{k} \leq \mathsf{a} \to \mathsf{k} = \mathsf{a} \ \vee \ \mathsf{k+1} \leq \mathsf{a}$	By Result 7

And since Q captures *less-than-or-equal-to*, we know it proves $\mathsf{k} \leq \mathsf{k+1}$, hence

iv. $\mathsf{a} = \mathsf{k} \to \mathsf{a} \leq \mathsf{k+1}$

But (i) to (iv) immediately entail

v. $\mathsf{a} \leq \mathsf{k+1} \ \vee \ \mathsf{k+1} \leq \mathsf{a}$

Since a is arbitrary, generalizing gives us what we needed to show. ☒

7.3 Defining the Δ_0, Σ_1 and Π_1 wffs

(a) Why did we bother to prove all those slightly tedious elementary results? Because they enable us easily to show that Q can prove *every* truth in the class of so-called Σ_1 wffs.

But why do we care about that class of wffs (whatever it is)? Because later we'll also be able to show that these are just the wffs we need for expressing decidable properties or relations. So we'll be able to use the fact that Q copes with Σ_1 truths to prove that Q can indeed case-by-case capture each decidable property and relation and hence is sufficiently strong.

So what are these Σ_1 wffs? The headline news is that they are (equivalent to) *existentially quantified* wffs, where what follows one or more initial existential quantifiers is 'simple', i.e. lacks unbounded quantifiers.

More carefully, the 'simple' kernel is built up using the successor, addition, and multiplication functions, identity, the less-than-or-equal-to relation, plus the familiar propositional connectives and/or *bounded* quantification.[4] As we'll soon see, just as Q can prove all true equations and disprove all false ones, it can also correctly decide all 'simple' wffs about particular numbers. So it can also prove the existential quantifications of the true ones. Which is why Q can prove all the true Σ_1 wffs.

So much for the headline news. In the rest of this section we'll pause to define the 'simple' wffs carefully: more officially, these are the Δ_0 wffs. Then we define the Σ_1 wffs which existentially quantify 'simple' wffs, and also the Π_1 wffs which universally quantify them.[5] And in the next section we'll show that Q can indeed prove all true Σ_1 sentences. You are very welcome to skim and skip.

(b) Recall, a *term* of L_A is an expression built up from '0' and/or variables by zero or more applications of the successor, addition and/or multiplication functions (see Section 4.3, (a)). We then say:

i. An *atomic* Δ_0 wff is a wff of one of the forms $\sigma = \tau$, $\sigma \leq \tau$, where σ and τ are terms.

So $\mathsf{S0} \leq \mathsf{S000}$, $\mathsf{x} = \mathsf{S000} + \mathsf{y}$, and $\mathsf{SS(y \times S0)} = \mathsf{y \times y}$, are examples of atomic Δ_0 wffs. And now the full class of Δ_0 wffs can be defined as follows:

[4] 'Hold on! What is meant by "bounded" quantification here? After all, a bounded quantification like $(\exists \mathsf{x} \leq 2)\mathsf{Fx}$ is just short for $\exists \mathsf{x}(\mathsf{x} \leq 2\ \wedge \mathsf{Fx})$, which involves an ordinary quantifier, running over all the domain. So, when abbreviations are unpacked, all quantifiers are on a par.' Well, in one sense, that's true enough! So let's be more careful and say that an existential quantifier, say $\exists \mathsf{x}$, has a bounded occurrence when it occurs in a subformula of the form $\exists \mathsf{x}(\mathsf{x} \leq \kappa \wedge \varphi(\mathsf{x}))$, for some numeral or variable κ and some open wff $\varphi(\mathsf{x})$ with at least 'x' free. Similarly, a universal quantifier, say $\forall \mathsf{x}$, has a bounded occurrence when it occurs in a subformula of the form $\forall \mathsf{x}(\mathsf{x} \leq \kappa\ \rightarrow\ \varphi(\mathsf{x}))$. Then the idea is that a 'simple' wff, if it involves quantifiers at all, involves only quantifiers with bounded occurrences.

[5] The 'Σ' in the standard label 'Σ_1' comes from an old alternative notation for the existential quantifier, as in $\Sigma x F x$. Likewise the 'Π' in 'Π_1' comes from corresponding notation for the universal quantifier, as in $\Pi x F x$. And the subscript '1' in 'Σ_1' and 'Π_1' indicates that we are dealing with wffs which start with *one* block of similar quantifiers, respectively existential quantifiers and universal quantifiers.

In this book, we won't be much concerned with more complex wffs. But for the record, in the same notation, a (strictly) Σ_n wff starts with n blocks of quantifiers of alternating types, a block of existential quantifiers followed by a block of universals followed by a block of existentials, etc, followed by a 'simple' kernel. Likewise a Π_n wff begins with n blocks of quantifiers of alternating type, but with a block of universals first.

Note that a Σ_0 wff will therefore be one in which a 'simple' kernel is preceded by no quantifiers at all – so the Σ_0 wffs are none other than the Δ_0 wffs (as we've dubbed them). And in fact both labels are current.

ii. 1) Every atomic Δ_0 wff is a Δ_0 wff;

2) If φ and ψ are Δ_0 wffs, so are $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ and $(\varphi \leftrightarrow \psi)$ (assuming all those connectives are either basic or defined in L_A).

3) If φ is a Δ_0 wff, so are $(\forall\xi \leq \kappa)\varphi$ and $(\exists\xi \leq \kappa)\varphi$, where ξ is any variable free in φ, and κ is a numeral or a variable distinct from ξ.[6]

4) Nothing else is a Δ_0 wff.

Hence, e.g., $(\exists \mathsf{y} \leq \mathsf{S000})(\mathsf{x} = \mathsf{S0} + \mathsf{y})$ and $\neg(\forall \mathsf{y} \leq \mathsf{x})(\mathsf{y} \leq \mathsf{S0} + \mathsf{z})$ are Δ_0 wffs.

Evidently, we can work out the truth or falsity of any *closed* Δ_0 sentence by a simple mechanical calculation, because we only have to look through a finite number of cases when we have to deal with a bounded quantification.

(c) That defines the 'simple' wffs. Then, as we said, Σ_1 wffs are wffs equivalent to (unbounded) existential quantifications of simple wffs. So we can continue our bunch of definitions as follows:

iii. A wff is strictly Σ_1 if it is of the form $\exists\xi\exists\zeta \ldots \exists\eta\varphi$, where φ is Δ_0 and $\xi, \zeta, \ldots, \eta$ are one or more distinct variables free in φ. A wff is Σ_1 if it is logically equivalent to a strictly Σ_1 wff.

Finally, for future use, here's a parallel definition:

iv. A wff is strictly Π_1 wff if it is of the form $\forall\xi\forall\zeta \ldots \forall\eta\varphi$ where φ is Δ_0. A wff is Π_1 if it is logically equivalent to a strictly Π_1 wff.

(d) Three mini-facts are worth noting immediately:

1. The negation of a Δ_0 wff is also Δ_0.
2. The negation of a Σ_1 wff is Π_1, and the negation of a Π_1 wff is Σ_1.
3. A Δ_0 wff is also both Σ_1 and Π_1.

The first is just trivial. The second is almost as trivial given the familiar equivalence of '$\neg\exists\mathsf{x}$' with '$\forall\mathsf{x}\neg$', etc. And for the third fact, suppose that the Δ_0 wff φ doesn't have e.g. the variable 'z' free. Then $(\varphi \wedge \mathsf{z} = \mathsf{z})$ is also Δ_0, and $\exists\mathsf{z}(\varphi \wedge \mathsf{z} = \mathsf{z})$ is strictly Σ_1 and $\forall\mathsf{z}(\varphi \wedge \mathsf{z} = \mathsf{z})$ is strictly Π_1. But each of those is logically equivalent to the original φ; hence φ also counts as both Σ_1 and Π_1.

7.4 Q is Σ_1 complete

Two more definitions. Suppose Γ is some class of wffs, and T is an interpreted theory. Then we say

[6] Why that distinctness requirement? Because, e.g., $(\forall \mathsf{x} \leq \mathsf{x})\varphi(\mathsf{x}) =_{\text{def}} \forall\mathsf{x}(\mathsf{x} \leq \mathsf{x} \rightarrow \varphi(\mathsf{x}))$ is trivially equivalent to $\forall\mathsf{x}\varphi(\mathsf{x})$, which is unbounded and so not what we want.

i. T is Γ-*sound* if, whenever $\varphi \in \Gamma$ and $T \vdash \varphi$, then φ is true.

ii. T is Γ-*complete* if, whenever $\varphi \in \Gamma$ and φ is true, then $T \vdash \varphi$.

(Here, 'true' of course means true on the standard interpretation built into T.)

With that jargon to hand, we can state the following simple but very important theorem:

> **Theorem 9** Q *is* Σ_1 *complete.*

And we can demonstrate this by proving in turn that

1. Q correctly decides every atomic Δ_0 sentence.[7]
2. Q correctly decides every Δ_0 sentence.
3. Q proves every true Σ_1 sentence.

We already know that Q can correctly decide every quantifier-free sentence (i.e. every sentence it shares with BA: see the opening remark of Section 6.4). So (2) extends that simple result to cover sentences with bounded quantifiers.

Proof for (1) An atomic Δ_0 *sentence* – i.e. a Δ_0 wff without variables – is either an equation $\tau_1 = \tau_2$ or else a wff of the form $\tau_1 \leq \tau_2$, where τ_1 and τ_2 are closed terms. If the first, we are done, because we know that Q correctly decides every equation. If the second, again because Q correctly decides every equation, we know Q can prove a couple of wffs correctly evaluating the terms, i.e. can prove $\tau_1 = \mathsf{t}_1$ and $\tau_2 = \mathsf{t}_2$ with numerals on the right. But since '$\leq$' captures the less-than-or-equal-to relation, Q correctly decides whether $\mathsf{t}_1 \leq \mathsf{t}_2$. Hence, plugging in the identities, Q correctly decides whether $\tau_1 \leq \tau_2$. ⊠

Proof for (2) Let's say that a Δ_0 sentence has degree k if it is built up from atomic wffs by k applications of connectives and/or bounded quantifiers.

The degree 0 sentences are the atomic sentences, and we now know that *they* are correctly decided by Q. So let's assume Q correctly decides all Δ_0 sentences of degree up to k. We'll show that it correctly decides χ, an arbitrary degree $k+1$ sentence. There are three cases to consider.

(i) χ is built using a propositional connective from φ and/or ψ, sentences of lower degree which by assumption Q correctly decides. But by elementary logic, if Q correctly decides φ and ψ, it correctly decides $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$ and $(\varphi \leftrightarrow \psi)$. And so Q correctly decides χ.

(iia) χ is of the form $(\forall\xi \leq \mathsf{n})\varphi(\xi)$.[8] If χ is a *true* sentence, then $\varphi(\mathsf{0}), \varphi(\mathsf{1}), \ldots, \varphi(\mathsf{n})$ must all be true sentences. Being of lower degree, these are – by hypothesis – all correctly decided by Q; so Q proves $\varphi(\mathsf{0}), \varphi(\mathsf{1}), \ldots, \varphi(\mathsf{n})$. Hence, by Result

[7] We defined 'correctly decides' in Section 3.4. Q correctly decides φ just in case, if φ is true, $\mathsf{Q} \vdash \varphi$, and if φ is false, $\mathsf{Q} \vdash \neg\varphi$.

[8] Given χ is a sentence, it can't be of the form $(\forall\xi \leq \nu)\varphi(\xi)$ with ν a variable dangling free.

4 of Section 7.2, we can conclude $(\forall\xi \leq \mathsf{n})\varphi(\xi)$. On the other hand, if χ is *false*, $\varphi(\mathsf{k})$ is false for some $k \leq n$, and – being of lower degree – this is correctly decided by Q, so Q proves $\neg\varphi(\mathsf{k})$. Hence, by Result 5, Q proves $(\exists\mathsf{x} \leq \mathsf{n})\neg\varphi(\mathsf{x})$, which easily gives us $\neg(\forall\mathsf{x} \leq \mathsf{n})\neg\varphi(\mathsf{x})$. So Q correctly decides χ, i.e. $(\forall\xi \leq \mathsf{n})\varphi(\xi)$.

(iib) χ is of the form $(\exists\xi \leq \mathsf{n})\varphi(\xi)$. Dealt with similarly to case (iia).

In sum, Q correctly decides all Δ_0 sentences of degree 0; also, if it decides all Δ_0 sentences of degree up to k, it decides all sentences of degree up to $k+1$. Therefore, by an informal induction on k, it decides all Δ_0 sentences, whatever their degree. ⊠

Proof for (3) Take, for example, a strictly Σ_1 sentence of the type $\exists\mathsf{x}\exists\mathsf{y}\varphi(\mathsf{x},\mathsf{y})$, where $\varphi(\mathsf{x},\mathsf{y})$ is Δ_0. If this sentence is true, then for some pair of numbers $\langle m, n\rangle$, the Δ_0 sentence $\varphi(\mathsf{m},\mathsf{n})$ must be true. Therefore, by (2), Q proves $\varphi(\mathsf{m},\mathsf{n})$. Hence $\exists\mathsf{x}\exists\mathsf{y}\varphi(\mathsf{x},\mathsf{y})$, by existential introduction. Evidently the argument generalizes for any number of initial quantifiers, which shows that Q proves all true strictly Σ_1 sentences. So it will prove their logical equivalents too. ⊠

7.5 An intriguing corollary

Let's say a theory T extends Q if it can prove all Q's theorems and retains the same interpretation for those theorems.[9] Since Q is Σ_1 complete, so is any theory T which extends Q.

It immediately follows that

Theorem 10 *If T extends* Q*, T is consistent iff it is* Π_1*-sound.*

Proof T is Π_1-sound if every Π_1 wff that T proves is true. First, then, suppose T proves a *false* Π_1-wff φ. $\neg\varphi$ will then be a *true* Σ_1 wff. But since T extends Q and so is Σ_1 complete, T will also prove $\neg\varphi$, making T inconsistent. Contraposing, if T is consistent, it proves no false Π_1-wff.

The converse is trivial, since if T is inconsistent, it proves anything, including false Π_1 wffs and so isn't Π_1-sound. ⊠

This is, in its way, a remarkable observation. It means that we don't have to fully *believe* a theory T – accept *all* its theorems are true – in order to use it to establish that some Π_1 generalization is true. We just have to believe that T is a consistent theory which extends Q.

Here's an imaginary example. Like many interesting arithmetical claims, Goldbach's conjecture that every even number greater than two is the sum of two primes can be expressed by a Π_1 sentence.[10] As we've noted before, no proof of

[9] Perhaps T has a richer language than L_A; but at least in the region of overlap, it gives the same interpretation to the shared wffs.

[10] Why so? Well, the property of being even can be expressed by the Δ_0 wff

$$\psi(\mathsf{x}) =_{\text{def}} (\exists\mathsf{v} \leq \mathsf{x})(2 \times \mathsf{v} = \mathsf{x})$$

And the property of being prime can be expressed by the Δ_0 wff the conjecture is currently known. But now suppose that some really ingenious deduction of it is found within e.g. Zermelo-Fraenkel set theory plus the negation of the Axiom of Choice. It doesn't matter for the current argument that you understand what this set theory says. All you only need to know that (i) it is certainly strong enough to define arithmetical vocabulary and to prove what Q can prove but also (ii) few people think that this is the 'right' set theory (whatever exactly that amounts to). Still it just doesn't matter whether or not we regard ZF + ¬AC as 'right'. So long as we accept – as pretty much everyone does – that this theory is *consistent*, a demonstration that it entails Goldbach's conjecture would be enough to establish that the conjecture is true-in-arithmetic.

We'll return to consider the force of this observation later, when we touch on Hilbert's programme again (cf. Section 1.6). But we'll finish this chapter by noting a corollary of Theorem 9, which is initially even more surprising:

> **Theorem 11** *Suppose T is a theory that extends* Q*, and φ is a Π_1 sentence such that T doesn't decide φ. Then φ is true.*

Proof If T doesn't decide φ, and so in particular we don't have $T \vdash \neg\varphi$, then the Σ_1 wff $\neg\varphi$ can't be true (for if it were true, T would prove it, since T extends Q and so is Σ_1-complete). But if $\neg\varphi$ can't be true, φ *is* true. ⊠

$$\chi(\mathsf{x}) =_{\text{def}} \mathsf{x} \neq 1 \ \wedge \ (\forall \mathsf{u} \leq \mathsf{x})(\forall \mathsf{v} \leq \mathsf{x})(\mathsf{u} \times \mathsf{v} = \mathsf{x} \ \rightarrow \ (\mathsf{u} = 1 \ \vee \ \mathsf{v} = 1))$$

where we rely on the trivial fact that a number's factors can be no greater than it. Then we can express Goldbach's conjecture as

$$\forall \mathsf{x}\{(\psi(\mathsf{x}) \wedge 4 \leq \mathsf{x}) \rightarrow (\exists \mathsf{y} \leq \mathsf{x})(\exists \mathsf{z} \leq \mathsf{x})(\chi(\mathsf{y}) \wedge \chi(\mathsf{z}) \wedge \mathsf{y} + \mathsf{z} = \mathsf{x})\}$$

which is Π_1 since what is after the initial quantifier is Δ_0.

8 First-order Peano Arithmetic

We put quite a bit of effort into discussing what Q can prove because (as we said) it will turn out that Q – and therefore any richer theory – is 'sufficiently strong' in the sense of Section 5.1. But still, Q is in other ways a very weak theory. And to derive elementary general truths like $\forall \mathsf{x}(0 + \mathsf{x} = \mathsf{x})$ that are beyond Q's reach, we evidently need a formal arithmetic that incorporates some stronger axiom(s) for proving quantified wffs. This chapter discusses the theory PA that we get if we add the most natural *induction axioms* to Q. We also briefly touch on a sub-theory of PA called Presburger Arithmetic.

8.1 Induction and the Induction Schema

(a) In informal argumentation, we frequently appeal to the following *principle of mathematical induction* in order to prove general claims:

> Suppose (i) 0 has the numerical property P. And suppose (ii) for any number n, if it has P, then its successor $n + 1$ also has P. Then we can conclude that (iii) *every* number has property P.[1]

In fact, we used informal inductions a number of times in the last chapter. For example, to prove Result 8 in Section 7.2, we in effect said: let n have property P if Q proves $\vdash \forall \mathsf{x}(\mathsf{x} \leq \mathsf{n} \lor \mathsf{n} \leq \mathsf{x})$. Then we argued (i) 0 has property P, and (ii) for any number n, if it has P, then $n + 1$ also has P. So we concluded (iii) every number has P, i.e. Q proves that wff for every n.

Why are such inductive arguments good arguments? Well, suppose (i) and (ii) hold. By (i) 0 has P. By (ii), if 0 has P so does $S0$. Hence $S0$ has P. By (ii) again, if $S0$ has P so does $SS0$. Hence $SS0$ has P. Likewise, $SSS0$ has P. And so on and so forth, through all the successors of 0. But the successors of 0 are the only natural numbers. So *all* natural numbers have property P. The intuitive induction principle is therefore underwritten by the basic structure of

[1] Mathematicians will be familiar with other ways of stating an induction principle. For example, there is 'course of values' induction, which says that if (i) 0 has property P and (ii$'$) the supposition that *every* number less than n has P implies that n has P, then (iii) every number has property P. And there is 'the method of infinite descent', which is the principle that if (iv) the supposition that n has P' always implies there is a smaller number $m < n$ such that m has P', then (v) no number has P'. It is an elementary exercise to show that these informal principles are equivalent.

We are going to be adding to Q a schema that, in a natural way, reflects the principle of induction as we've stated it above. We could, alternatively, add a schema that reflects course of values induction or the method of infinite descent. We'd end up with exactly equivalent formal theories. Which is why we're not going to bother to mention these alternatives any further.

the number sequence, and in particular by the absence of ‘stray’ numbers that you can’t get to step-by-step from zero.

(b) Now, our intuitive principle is naturally interpreted as a generalization covering any genuine property of numbers P. Hence to frame a corresponding formal version, we’d ideally like to use a language that enables us to generalize over all numerical properties. But, as we announced at the very beginning of Chapter 6, we are currently concentrating on formal theories built in languages like L_A whose logical apparatus involves no more than the familiar first-order quantifiers that range over the domain of numbers: we don’t have second-order quantifiers available to range over properties of numbers. So how can we handle induction?

Our only option is to use a schema again.[2] As a first shot, then, let’s try the following: we’ll say that *any closed wff that is an instance of the Induction Schema*

$$(\{\varphi(\mathsf{0}) \wedge \forall \mathsf{x}(\varphi(\mathsf{x}) \rightarrow \varphi(\mathsf{Sx}))\} \rightarrow \forall \mathsf{x}\varphi(\mathsf{x}))$$

is to count as an axiom, where $\varphi(\mathsf{x})$ stands in for some open wff of L_A with just ‘x’ free (and $\varphi(\mathsf{0})$ and $\varphi(\mathsf{Sx})$ are, of course, the results of systematically substituting ‘0’ and ‘Sx’ respectively for ‘x’).

Since $\varphi(\mathsf{x})$ will be constructed from no more than the constant term ‘0’, the successor, addition and multiplication functions, plus identity and other logical apparatus, it surely will express a perfectly determinate arithmetical property. Hence the intuitive principle will apply, and so the relevant instance of the Induction Schema will be true.

(c) So far, so good. But we now need to generalize a bit, because we will want to use inductive arguments to prove general results about *relations* as well as about monadic properties. How can we handle these?

Let’s take a simple illustrative example. Suppose ‘Rxyz’, for example, abbreviates some open wff L_A wff with three free variables, which expresses the relation R. Let’s arbitrarily pick objects a and b from the domain. And suppose just for a moment that our logical apparatus allows these to be temporarily named ‘a’ and ‘b’. Then e.g. ‘Rxab’ now expresses a monadic property – the property of standing in the R relation to a and b. So the intuitive induction principle will again apply to this property as before. Hence the following will be true (where rogue brackets are added for readability):

$$(\{\mathsf{R0ab} \wedge \forall \mathsf{x}(\mathsf{Rxab} \rightarrow \mathsf{R(Sx)ab})\} \rightarrow \forall \mathsf{xRxab})$$

But we said that ‘a’ and ‘b’ denote arbitrary elements of the domain. Hence we can generalize into the places temporarily held by these names, to get

[2] Compare BA where we had to use schemas because we then couldn’t even quantify over *objects*: now we are using a schema because even in a full first-order language we can’t quantify over *properties*.

$$\forall\mathsf{y}\forall\mathsf{z}(\{\mathsf{R0yz} \land \forall\mathsf{x}(\mathsf{Rxyz} \to \mathsf{R(Sx)yz})\} \to \forall\mathsf{xRxyz})$$

and this proposition is still warranted by our intuitive principle.

Now, what we have just arrived at is the *universal closure* of the result of plugging $\varphi(\mathsf{x}) = \mathsf{Rxyz}$ into our original Induction Schema: i.e. it's what you get when you prefix universal quantifiers to bind the free variables left in the instance of the Schema for Rxyz. And *we've shown that this universal closure is intuitively warranted as a new induction axiom.*

(d) What goes for Rxyz will go quite generally. So this motivates the following more expansive way of characterizing the family of intuitively sound induction axioms expressible in L_A:

> Any sentence that is the universal closure of an instance of the
>
> **Induction Schema** $(\{\varphi(\mathsf{0}) \land \forall\mathsf{x}(\varphi(\mathsf{x}) \to \varphi(\mathsf{Sx}))\} \to \forall\mathsf{x}\varphi(\mathsf{x}))$
>
> is to count as an axiom, where $\varphi(\mathsf{x})$ is now an open wff that may also have variables other than 'x' free.

(e) One quick comment before proceeding. Much later, we will in fact consider arithmetics with a second-order Induction Axiom which *does* quantify over all properties of numbers. But just what counts as 'all'?

At the generous end of the spectrum of possible views here, we might hold – naturally enough – that *any* arbitrary set of numbers Σ corresponds to a genuine monadic property P_Σ (where n has the property P_Σ if and only if $n \in \Sigma$). But we know from Theorem 1, comment (b), that there are non-enumerably many such sets of numbers Σ. So, on the generous view, a second-order Induction Axiom covers non-enumerably many monadic properties.

However, the open wffs $\varphi(\mathsf{x})$ of L_A (or of any properly formalized language) are by contrast enumerable. So, the first-order Induction Schema covers only enumerably many monadic properties. Hence we will expect an arithmetic that uses the first-order Schema to be notably weaker than one that uses the full second-order Axiom (at least if we take the generous view about properties). This expectation will be confirmed in due course.

8.2 PA – First-order Peano Arithmetic

Our discussion of induction in the last section motivates moving on from Q – and jumping right over a range of intermediate theories[3] – to adopt the much richer formal theory of arithmetic that we can briskly define as follows:

[3] For a discussion of some intermediate theories of arithmetic with various restrictions on the Induction Schema, e.g. with induction only allowed for Δ_0 or Σ_1 wffs, see the wonderful (Hájek and Pudlák, 1993). These theories are technically very interesting, but are not for the moment particularly relevant for us, so we won't delay over them.

> PA – First-order Peano Arithmetic[4] – is the result of adding to the axioms of Q the universal closures of all instances of the Induction Schema.

Plainly, it is decidable whether any given wff has the right shape to be one of the new axioms, so PA is a legitimate formalized theory.

Let's immediately give three quick and easy examples of induction in action.

(a) First consider the wff $\forall x(0 + x = x)$. We showed in Section 6.4 that this is unprovable in Q. But we can prove it in PA by induction.

To establish this, we'll again assume a Fitch-style natural deduction logic, with the standard rules UI (Universal Instantiation), UG (Universal Generalization on 'arbitrary' names) and CP (Conditional Proof). And to derive our target wff $\forall x(0 + x = x)$, we will need to start with an instance of the Induction Schema with $\varphi(x)$ replaced by $(0 + x = x)$. Then we aim to prove the two conjuncts in the antecedent of that instance, so we can extract the desired conclusion by a final modus ponens. Here's a formal version:

1.	$(\{0 + 0 = 0 \ \wedge\ \forall x(0 + x = x \to 0 + Sx = Sx)\}$ $\to\ \forall x(0 + x = x))$	Instance of Induction
2.	$\forall x(x + 0 = x)$	Axiom 4
3.	$0 + 0 = 0$	From 2 by UI with 0
4.	$\forall x \forall y(x + Sy = S(x + y))$	Axiom 5
5.	$\quad 0 + a = a$	Supposition
6.	$\quad \forall y(0 + Sy = S(0 + y))$	From 4, by UI with 0
7.	$\quad 0 + Sa = S(0 + a)$	From 6 by UI with a
8.	$\quad 0 + Sa = Sa$	From 5, 7 by LL
9.	$0 + a = a \ \to\ 0 + Sa = Sa$	From 5 to 8 by CP
10.	$\forall x(0 + x = x \ \to\ 0 + Sx = Sx)$	From 9 by UG
11.	$\{0 + 0 = 0 \ \wedge\ \forall x(0 + x = x \ \to\ 0 + Sx = Sx)\}$	From 3, 10 by $\wedge$-intro.
12.	$\forall x(0 + x = x)$	From 10, 11 by MP

(b) Our deviant interpretation which makes the axioms of Q true while making $\forall x(0 + x = x)$ false has Gwyneth as a self-successor. The axioms of PA, however, rule out self-successors. Put $\varphi(x) =_{\mathrm{def}} (x \neq Sx)$. Then PA entails $\varphi(0)$ (that's Axiom 1), and entails $\forall x(\varphi(x) \to \varphi(Sx))$ (by contraposing Axiom 2). So by induction, we can conclude $\forall x \varphi(x)$, i.e. no number is a self-successor.[5]

(c) Next, more sketchily, here's how to prove $\forall x \forall y(x + y = y + x)$, another wff that is unprovable in Q (why?). Put $\varphi(x, y) =_{\mathrm{def}} (x + y = y + x)$, and note we have the following induction axiom:

[4] The name is conventional. Giuseppe Peano did indeed publish a list of axioms for arithmetic in Peano (1889). But they weren't first-order, only explicitly governed the successor relation, and – as he acknowledged – had already been stated by Richard Dedekind (1888).

[5] Don't be lulled into a false sense of security, though! While PA's axioms may rule out deviant interpretations based on self-successors, they don't rule out some other deviant interpretations, about which more anon.

$$\forall \mathsf{y}(\{\varphi(\mathsf{0},\mathsf{y}) \land \forall \mathsf{x}(\varphi(\mathsf{x},\mathsf{y}) \to \varphi(\mathsf{Sx},\mathsf{y}))\} \to \forall \mathsf{x}\varphi(\mathsf{x},\mathsf{y}))$$

Now, we've in effect established $\forall \mathsf{y}\varphi(\mathsf{0},\mathsf{y})$ in (a). So to prove our target wff, i.e. $\forall \mathsf{x}\forall \mathsf{y}\varphi(\mathsf{x},\mathsf{y})$, it is enough to show that $\forall \mathsf{x}\forall \mathsf{y}(\varphi(\mathsf{x},\mathsf{y}) \to \varphi(\mathsf{Sx},\mathsf{y}))$. And we *can* show this by another induction (change variables, put $\psi(\mathsf{x}) =_{\mathrm{def}} \forall \mathsf{u}(\varphi(\mathsf{u},\mathsf{x}) \to \varphi(\mathsf{Su},\mathsf{x}))$ and then use the instance of the Induction Schema for this open wff).

Enough! Induction proofs like these very rapidly get tedious. But then the point of the present exercise isn't user-friendliness but austere formal rigour: the game is to see what we really *need* by way of absolutely fundamental axioms in order to get standard arithmetical results. And a little investigation and experimentation should convince you at least of this much: PA does indeed have the resources to establish all the familiar elementary general truths about the addition and multiplication of numbers.

8.3 PA in summary

PA, then, is a much more powerful theory than Q, yet we have an entirely natural motivation for accepting its unrestricted use of the Induction Schema. For this reason, PA is the benchmark axiomatized first-order theory of basic arithmetic. Just for neatness, then, let's bring together all the elements of its specification in one place.

But first, a quick observation. Suppose we this time put

$$\varphi(\mathsf{x}) =_{\mathrm{def}} (\mathsf{x} \neq \mathsf{0} \to \exists \mathsf{y}(\mathsf{x} = \mathsf{Sy}))$$

Then $\varphi(\mathsf{0})$ is a trivial PA theorem. Likewise, $\forall \mathsf{x}\varphi(\mathsf{Sx})$ is also a trivial theorem, and that entails $\forall \mathsf{x}(\varphi(\mathsf{x}) \to \varphi(\mathsf{Sx}))$. So we can use an instance of the Induction Schema to derive $\forall \mathsf{x}\varphi(\mathsf{x})$. But that's just Axiom 3 of Q.[6] So our original presentation of PA – as explicitly having all the Axioms of Q plus the instances of the Induction Schema – involves a certain redundancy. Bearing that in mind, here's our summary overview:

First, the *language* of PA is L_A, a first-order language whose non-logical vocabulary comprises just the constant '0', the one-place function symbol 'S', and the two-place function symbols '+', '×', and whose given interpretation is the obvious one.

Second, PA's deductive *proof system* is some standard version of classical first-order logic with identity (differences between versions aren't significant).

And third, its *axioms* – eliminating the redundancy from our original statement of the axioms – are the following sentences (closed wffs)

[6] As we saw in Section 7.2, Axiom 3 enables us to prove some important general claims in Q, despite the absence of the full range of induction axioms. It, so to speak, functions as a very restricted surrogate for induction in certain proofs.

Axiom 1 $\forall\mathsf{x}(0 \neq \mathsf{Sx})$

Axiom 2 $\forall\mathsf{x}\forall\mathsf{y}(\mathsf{Sx} = \mathsf{Sy} \rightarrow \mathsf{x} = \mathsf{y})$

Axiom 3 $\forall\mathsf{x}(\mathsf{x} + 0 = \mathsf{x})$

Axiom 4 $\forall\mathsf{x}\forall\mathsf{y}(\mathsf{x} + \mathsf{Sy} = \mathsf{S}(\mathsf{x} + \mathsf{y}))$

Axiom 5 $\forall\mathsf{x}(\mathsf{x} \times 0 = 0)$

Axiom 6 $\forall\mathsf{x}\forall\mathsf{y}(\mathsf{x} \times \mathsf{Sy} = (\mathsf{x} \times \mathsf{y}) + \mathsf{x})$

plus every sentence that is the universal closure of an instance of the following

Induction Schema $(\{\varphi(0) \;\wedge\; \forall\mathsf{x}(\varphi(\mathsf{x}) \rightarrow \varphi(\mathsf{Sx}))\} \;\rightarrow\; \forall\mathsf{x}\varphi(\mathsf{x}))$

where $\varphi(\mathsf{x})$ is an open wff of L_A that has at least 'x', and perhaps other variables, free.

8.4 A very brief aside: Presburger Arithmetic

As we said, unlike the case with Q, exploration doesn't readily reveal any elementary and familiar arithmetical truths of L_A that PA can't prove. So we might reasonably have hoped – at least before we ever heard tell of Gödel's Theorems – that PA would turn out to be complete.

Here's another fact that might well have encouraged that hope, pre-Gödel. Suppose we define the language L_P to be L_A without the multiplication sign. Take P to be the theory couched in the language L_P, whose axioms are Q's now familar axioms for successor and addition, plus the universal closures of all instances of the Induction Schema that can be formed in L_P. So P is PA minus multiplication. *Then* P *is a negation-complete theory of successor and addition.*

We are not going to be able to prove that last claim in this book. The argument uses a standard model-theoretic method called 'elimination of quantifiers' which isn't hard, but it would just take too long to explain.[7] Note, though, that the availability of a complete theory for successor and addition was proved as early as 1929 by Mojzesz Presburger.[8]

So the situation is as follows, and was known before Gödel got to work. (i) There is a complete theory BA whose theorems are exactly the quantifier-free truths expressible using successor, addition and multiplication (and the connectives). (ii) There is a complete theory (in fact equivalent to PA minus multiplication) whose theorems are exactly the first-order truths expressible using just

[7] Enthusiasts will find an accessible outline of the proof in (Fisher, 1982, Ch. 7), which can usefully be read in conjunction with (Boolos et al., 2002, Ch. 24).

[8] To be strictly accurate, Presburger – then a young graduate student – proved the completeness not of P but of a rather different theory (whose primitive expressions were just '0', '1' and '+'). But a few years later, Hilbert and Bernays (1934) showed that his methods could be applied to the much neater theory P, and it is the latter which is these days typically referred to as 'Presburger Arithmetic'.

successor and addition. Against this background, Gödel's result that adding multiplication to get PA gives us a theory which is incomplete (and incompleteable) comes as a rather nasty surprise. Why on earth should adding multiplication make all the difference?[9]

8.5 Is PA consistent?

As we said, PA proves a great deal more than Q. But it wouldn't be much joy to discover that PA's strength is due to the theory's tipping over into being *inconsistent* and so entailing *every* wff. So let's finish this chapter by briefly considering the issue of consistency – not because we think that there's a sporting chance that PA might really be in trouble, but because it gives us an opportunity to mention themes that will occupy us later.

Take the given interpretation $\mathcal{I}_A$ which we built into PA's language $L_A = \langle \mathcal{L}_A, \mathcal{I}_A \rangle$. On this interpretation, '0' has the value zero; 'S' represents the successor function, etc. Hence, on $\mathcal{I}_A$, (1) the first two axioms of PA are evidently core truths about the operation that takes one number to its successor. And the next four axioms are equally fundamental truths about addition and multiplication. (2) We have already argued that the informal induction principle for arithmetical properties and relations is warranted by our understanding of the structure of the natural number sequence (and in particular, by the lack of 'stray' numbers outside the sequence of successors of zero). So since complex open wffs of L_A straightforwardly express genuine numerical properties and relations, (the closures of) all the instances of PA's Induction Schema will also be true on $\mathcal{I}_A$. But (3) the classical first-order deductive logic of PA is truth-preserving so – given that the axioms are true and PA's logical apparatus is in good order – all its theorems are true on $\mathcal{I}_A$. Hence (4), since all PA theorems are true on $\mathcal{I}_A$, there cannot be pairs of theorems of the form φ and $\neg\varphi$ (for these of course couldn't both be true together). So (5) the theory is consistent.

Now, this argument looks pretty compelling. But on second thoughts, how are we supposed to interpret it? Well, we could read the argument as an outline of a formal proof that we could set out in a background theory T rich enough to talk about wffs of PA and to incorporate a version of $\mathcal{I}_A$. The trouble is that if T is going to use this argument to show that the theorems of PA are all true, then T is going to have to be at least as rich as PA. Why? Well, if T incorporates $\mathcal{I}_A$, it can prove for example that '$\forall \mathsf{x}(0 \neq \mathsf{Sx})$' is true iff $\forall x(0 \neq Sx)$. But to go on to prove that PA's first Axiom actually *is* true, T will then itself need to prove $\forall x(0 \neq Sx)$. And so on through all the other axioms. But then, if there's any kind of issue about PA's consistency, there will be the same issue about T's consistency, and we haven't got anywhere.

[9] And by the way, it isn't that multiplication is in itself somehow intrinsically intractable. In 1931, Thoraf Skolem showed that there is a complete theory for the truths expressible in a suitable first order language with multiplication but lacking addition.

However, perhaps our sketched argument for PA's consistency is intended in a rather different way. Perhaps the idea isn't to invoke some further theory but to appeal more directly to our intuitive grasp of the natural numbers: we are supposed just to *see* that (1) and (2) – and hence (3) to (5) – are true. But then critics will want to emphasize the point that an argument for a theory's consistency which appeals to our supposed intuitive grasp of an intended interpretation *can* lead us badly astray. And to support their point, they will refer to one of the most famous episodes in the history of logic, which concerns the fate of the German logician Gottlob Frege's *The Basic Laws of Arithmetic*.[10]

Frege aimed to construct a formal system in which first arithmetic and then the theory of the real numbers can be rigorously developed by deducing them from logic-plus-definitions. He has a wide conception of what counts as logic, which embraces axioms for what is in effect a theory of classes,[11] so that the number sequence can be identified as a certain sequence of classes, and then rational and real numbers can be defined via appropriate classes of these classes. Frege takes as his fifth Basic Law the assumption, in effect, that *for every well-constructed open wff* $\varphi(\mathsf{x})$ *of his language, there is a class (possibly empty) of exactly those things that satisfy this wff.* And indeed, what could be more plausible? If we can coherently express some condition, then we should surely be able to talk about the (possibly empty) collection of just those things that satisfy that condition.

But, famously, the assumption is disastrous. As Bertrand Russell pointed out in a letter which Frege received as the second volume of *Basic Laws* was going through the press, the plausible assumption leads to contradiction.[12] Take for example the condition R expressed by '. . . is a class which isn't a member of itself'. This is, on the face of it, a perfectly coherent condition (the *class of people*, for example, satisfies the condition: the class of people contains only people, so it doesn't contain any classes, so doesn't contain itself in particular). And certainly condition R is expressible in the language of Frege's system. So on Frege's assumption, there will be a class of things that satisfy R. In other words, there is a class Σ_R of all the classes which aren't members of themselves. But now ask: is Σ_R a member of itself? A moment's reflection shows that it is if it isn't, and isn't if it is: contradiction! So there can be no such class as Σ_R; hence Frege's assumption cannot be right, despite its intuitive appeal, and his formal system which embodies that assumption is inconsistent.

This sad tale brings home to us vividly that intuitions of consistency can be mistaken. But let's not rush to make too much of this: the fact that we *can* make mistakes in arguing for the cogency of a formal system on the basis of our sup-

[10] The first volume of *Basic Laws* was published in 1893, the second in 1903. For a partial translation, see Frege (1964).

[11] When talking about the views of Frege and Russell, it seems more appropriate to use Russell's favoured term 'class' rather than 'set', if only because the latter has become so very closely linked to a specific post-Russellian idea, namely the iterative conception of sets (as explained, e.g., in Potter, 2004, §3.2).

[12] See (Russell, 1902).

posed grasp of an intended interpretation isn't any evidence that we *have* made a mistake in our argument for the consistency of PA. For a start, Peano Arithmetic and many stronger theories that embed it have been intensively explored for a century and no contradiction has been exposed.

'But can't we do better,' you might still ask, 'than make the negative point that no contradiction has been found (yet): can't we *prove* that PA is consistent in some other way than by going round in circles or by appealing to our supposed grasp of an interpretation?'

Yes, there *are* other proofs. However, we'll have to put further discussion of this intriguing issue on hold until after we have said more about Gödel's *Second* Incompleteness Theorem. For that Theorem is all about consistency proofs (see Section 1.5). It puts some interesting limits on the possibilities here. But that will all have to wait until Chapter 16.3.

9 Primitive recursive functions

The formal theories of arithmetic that we've looked at so far have (at most) the successor function, addition and multiplication built in. But why on earth stop there? School arithmetic acknowledges many more numerical functions. This chapter describes a very wide class of such functions, the so-called primitive recursive ones. Then in Chapter 11, we'll be able to show that Q and PA in fact already have the resources to deal with all these functions.

9.1 Introducing the primitive recursive functions

We'll start with two more functions that are familiar from elementary arithmetic. Take the factorial function $y!$, where e.g. $4! = 1 \times 2 \times 3 \times 4$. This can be defined by the following two equations:

$$\begin{aligned}&0! = S0 = 1\\&(Sy)! = y! \times Sy\end{aligned}$$

The first clause tells us the value of the function for the argument $y = 0$; the second clause tells us how to work out the value of the function for Sy once we know its value for y (assuming we already know about multiplication). So by applying and reapplying the second clause, we can successively calculate $1!$, $2!$, $3!$, ... Hence our two-clause definition fixes the value of '$y!$' for all numbers y.

For our second example – this time a two-place function – consider the exponential, standardly written in the form 'x^y'. This can be defined by a similar pair of equations:

$$\begin{aligned}&x^0 = S0\\&x^{Sy} = (x^y \times x)\end{aligned}$$

Again, the first clause gives the function's value for a given value of x and $y = 0$, and – keeping x fixed – the second clause gives the function's value for the argument Sy in terms of its value for y.

We've seen this two-clause pattern before, of course, in our formal Axioms for the multiplication and addition functions. Informally, and now presented in the style of everyday mathematics (i.e. without explicit quantifiers), we have:

$$\begin{aligned}&x \times 0 = 0\\&x \times Sy = (x \times y) + x\end{aligned}$$

$$\begin{aligned}&x + 0 = x\\&x + Sy = S(x + y)\end{aligned}$$

Three comments about our examples so far:

i. In each definition, an instance of the second clause can invoke the result of its own previous application (to a smaller number), a kind of procedure that is standardly termed 'recursive'. So this sort of two-clause definition of a function is called a *definition by recursion*.[1]

ii. Note, for example, that $(Sn)!$ is defined as $n! \times Sn$, so it is evaluated by evaluating $n!$ and Sn and then feeding the results of these computations into the multiplication function. This involves, in a word, the *composition* of functions, where evaluating a composite function involves taking the output(s) from one or more functions, and treating these as inputs to another function.

iii. Our series of examples illustrates two *chains* of definitions by recursion and functional composition. Working from the bottom up, addition is defined in terms of the successor function; multiplication is then defined in terms of successor and addition; and then the factorial (or on the other chain, exponentiation) is defined in terms of multiplication and successor.

Here's another little definitional chain:

$$P(0) = 0$$
$$P(Sx) = x$$

$$x \dot{-} 0 = x$$
$$x \dot{-} Sy = P(x \dot{-} y)$$

$$|x - y| = (x \dot{-} y) + (y \dot{-} x)$$

'P' signifies the predecessor function (with zero being treated as its own predecessor); '$\dot{-}$' signifies 'subtraction with cut-off', i.e. subtraction restricted to the non-negative integers (so $m \dot{-} n$ is zero if $m < n$). And $|m - n|$ is of course the absolute difference between m and n. This time, our third definition doesn't involve recursion, only a simple composition of functions.

These examples motivate the following initial gesture towards a definition:

> A *primitive recursive function* is one that can be similarly characterized by a chain of definitions by recursion and composition.[2]

That is a rather quick-and-dirty characterization, and it should be enough to get across the basic idea. However, we really need to pause to do better. In particular, we need to nail down more carefully the 'starter pack' of functions that we are allowed to take for granted in building a definitional chain.

[1] Strictly speaking, we need a proof of the claim that recursive definitions really do well-define functions: such a proof was first given by Richard Dedekind (1888, §126). Logic students, of course, are very familiar with a related kind of recursive definition, as illustrated e.g. by our definition of a 'term' in Section 4.3.

[2] The basic idea is there in Dedekind and highlighted by Thoralf Skolem (1923). But the modern terminology 'primitive recursion' seems to be due to Rósza Péter (1934); and 'primitive recursive function' was first used in Stephen Kleene's classic (1936a).

9.2 Defining the p.r. functions more carefully

(a) Consider the recursive definition of the factorial again:

$$\begin{aligned}&0! = 1\\&(Sy)! = y! \times Sy\end{aligned}$$

This is an example of the following general scheme for defining a one-place function f:

$$\begin{aligned}&f(0) = g\\&f(Sy) = h(y, f(y))\end{aligned}$$

Here, g is just a number, while h is – crucially – a function we are assumed already to know about prior to the definition of f (maybe because h is an 'initial' function that we are allowed to take for granted like the successor function; or perhaps because we've already given recursion clauses to define it; or perhaps because it is a composite function constructed by plugging one known function into another – as in the case of the factorial, where $h(y, u) = u \times Sy$).

Likewise, with a bit of massaging, the recursive definitions of addition, multiplication and the exponential can all be treated as examples of the following general scheme for defining two-place functions:

$$\begin{aligned}&f(x, 0) = g(x)\\&f(x, Sy) = h(x, y, f(x, y))\end{aligned}$$

where now g and h are both functions that we already know about. Three points about this:

i. To get the definition of addition to fit this pattern, we have to take $g(x)$ to be the trivial identity function $I(x) = x$.

ii. To get the definition of multiplication to fit the pattern, $g(x)$ has to be treated as the even more trivial zero function $Z(x) = 0$.

iii. Again, to get the definition of addition to fit the pattern, we have to take $h(x, y, u)$ to be the function Su. As this illustrates, we must allow h not to care what happens to some of its arguments. One neat way of doing this is to help ourselves to some more trivial identity functions that serve to select out particular arguments. Suppose, for example, we have the three-place function $I(x, y, u) = u$ to hand. Then, in the definition of addition, we can put $h(x, y, u) = SI(x, y, u)$, so h is defined by composition from previously available functions.

So with that motivation, we will now officially say that the full 'starter pack' of *initial functions* contains, as well as the successor function S, the boring zero function $Z(x) = 0$ and all the k-place identity functions, $I_i^k(x_1, x_2, \ldots, x_k) = x_i$ for each k, and for each i, $1 \leq i \leq k$.[3]

[3] The identity functions are also often called *projection* functions. They 'project' the vector with components $x_1, x_2, \ldots, x_k$ onto the i-th axis.

(b) We next want to generalize the idea of recursion from the case of one-place and two-place functions. There's a (standard) notational device that helps to put things snappily: we'll henceforth write $\vec{x}$ as short for the array of k variables $x_1, x_2, \ldots, x_k$. Then we can generalize as follows:

> Suppose that the following holds:
>
> $$f(\vec{x}, 0) = g(\vec{x})$$
> $$f(\vec{x}, Sy) = h(\vec{x}, y, f(\vec{x}, y))$$
>
> Then f *is defined from* g *and* h *by recursion.*

If we allow $\vec{x}$ to be empty, so $g(\vec{x})$ is a constant, that subsumes the case of one-place functions like the factorial.

(c) Finally, we need to tidy up the idea of definition by composition. The basic idea, to repeat, is that we form a composite function f by treating the output value(s) of one or more given functions g, g', g'' as the input argument(s) to another function h. For example, we set $f(x) = h(g(x))$. Or, to take a slightly more complex case, we set $f(x, y, z) = h(g(x, y), g'(y, z))$.

There's a number of equivalent ways of covering the manifold possibilities of compounding multi-place functions. But one standard way is to define what we might call one-at-a-time composition (where we just plug one function g into another function h), thus:

> If $g(\vec{y})$ and $h(\vec{x}, u, \vec{z})$ are functions – with $\vec{x}$ and $\vec{z}$ possibly empty – then f *is defined by composition by substituting* g *into* h just if $f(\vec{x}, \vec{y}, \vec{z}) = h(\vec{x}, g(\vec{y}), \vec{z})$.

We can then think of generalized composition – where we plug more than one function into another function – as just iterated one-at-a-time composition. For example, we can substitute the function $g(x, y)$ into $h(u, v)$ to define the function $h(g(x, y), v)$ by composition. Then we can substitute $g'(y, z)$ into the defined function $h(g(x, y), v)$ to get the composite function $h(g(x, y), g'(y, z))$

(d) We informally defined the primitive recursive functions as those that can be defined by a chain of definitions by recursion and composition. Working backwards down a definitional chain, it must bottom out with members of an initial 'starter pack' of trivially simple functions. At the start of the chapter, we highlighted the successor function among the given simple functions. But we've since noted that, to get our examples to fit our official account of definition by primitive recursion, we'll have to acknowledge some other, even more trivial, initial functions.

Putting everything together, we can now offer this more formal characterization of the p.r. functions (as we'll henceforth call them for short):[4]

[4] Careful! Some books use 'p.r.' to abbreviate 'partial recursive', which is a quite different idea we'll meet much later. Our abbreviatory usage is, however, the more common one.

1. The initial functions S, Z, and I_i^k are p.r.;
2. if f can be defined from the p.r. functions g and h by composition, substituting g into h, then f is p.r.;
3. if f can be defined from the p.r. functions g and h by primitive recursion, then f is p.r.;
4. nothing else is a p.r. function.

Note, by the way, that the initial functions are *total* functions of numbers (i.e. are defined for every numerical argument); and that primitive recursion and composition both build total functions out of total functions. Which means that all p.r. functions are total functions, defined for all natural number arguments.

9.3 An aside about extensionality

We'd better pause for a clarificatory aside, a general point about the identity conditions for functions, which is then applied to p.r. functions in particular.

If f and g are one-place total numerical functions, we count them as being the *same* function iff, for each n, $f(n) = g(n)$. More generally, we count f and g as the same function iff they have the same extension, i.e. just so long as they pair up arguments with values in the same way. In a word, we construe talk of functions *extensionally*.[5]

Of course, one and the same function can be presented in different ways, e.g. in ways that reflect different rules for calculating it. For a trivial example, the function $2n+1$ is the same function as $(n+1)^2 - n^2$; but the two different modes of presentation indicate different routines for evaluating the function.

Now, a p.r. function, in particular, is by definition one that *can* be specified by a certain sort of chain of definitions. And so the natural way of presenting one will be by giving a definitional chain for it (and thereby making it transparent that the function is indeed p.r.). But the same function can be presented in other ways; and some modes of presentation can completely disguise the fact that the given function is recursive. For a dramatic example, consider the function

> $\mathit{fermat}(n) = n$ if there are solutions to $x^{n+3} + y^{n+3} = z^{n+3}$ (with x, y, z positive integers);
>
> $\mathit{fermat}(n) = 0$ otherwise.

This definition certainly doesn't reveal whether the function is primitive recursive. But we know now – thanks to Andrew Wiles's proof of Fermat's Last

[5] Compare Section 4.2 where we said that P and Q as the same property if they have the same extension. If you accept the thesis of Frege (1891), then we have to treat properties and functions in the same way. For Frege urges us to treat properties as just a special kind of function – so a numerical property, in particular, is a function that maps a number to the truth-value *true* (if the number has the property) or *false* (otherwise) – which comes very close to identifying a property with its characteristic function.

Theorem – that *fermat* is in fact p.r., for it is none other than (i.e. has the same extension as) the trivially p.r. function $Z(n) = 0$.

Note too that other modes of presentation may make it clear that a function is p.r., but still not tell us *which* p.r. function is in question. Consider, for example, the function defined by

$j(n) = n$ if Julius Caesar ate grapes on his third birthday;
$j(n) = 0$ otherwise.

There is no way (algorithmic or otherwise) of settling what Caesar ate on his third birthday! But despite that, the function $j(n)$ is plainly primitive recursive. Why so? Well, either it is the trivial identity function $I(n) = n$, or it is the zero function $Z(n) = 0$. So we know that $j(n)$ must be a p.r. function, though we can't determine *which* function it is from this mode of presentation.

In sum, primitive recursiveness is a feature of the function itself (identified extensionally), not a feature of its mode of presentation.

9.4 The p.r. functions are computable

To repeat, a p.r. function f must be specifiable by a chain of definitions by recursion and composition leading back ultimately to initial functions. But (a) the initial functions S, Z, and I_i^k are trivially computable. (b) The composition of two computable functions g and h is computable (you just feed the output from whatever computer routine evaluates g as input into the routine that evaluates h). And (c) – the key point – if g and h are computable, and f is defined by primitive recursion from g and h, then f is computable too. So as we build up longer and longer chains of definitions for p.r. functions, we always stay within the class of computable functions.

To illustrate (c), return once more to our example of the factorial. Here's its p.r. definition again:

$$0! = 1$$
$$(Sy)! = y! \times Sy$$

The first clause gives the value of the function for the argument 0; then you can repeatedly use the second recursion clause to calculate the function's value for $S0$, then for $SS0$, $SSS0$, etc. So the definition encapsulates an algorithm for calculating the function's value, and corresponds exactly to a certain simple kind of computer routine.

Thus compare our definition with the following schematic program:

```
1. $fact := 1$
2. For $y = 0$ to $n - 1$
3.     $fact := (fact \times Sy)$
4. Loop
```

Here *fact* is a memory register that we initially prime with the value of 0!. Then the program enters a loop: and the crucial thing about a 'for' loop is that the total number of iterations is fixed in advance. The program numbers the loops from 0, and on loop number k the program replaces the value in the register with Sk times the previous value: we'll assume the computer already knows how to find the successor of k and do that multiplication. When the program exits the loop after a total of n iterations, the value in the register *fact* will be $n!$.

Generalizing, for a one-place function f defined by recursion in terms of g and the computable function h, the same program structure always does the trick for calculating $f(n)$. Thus compare

$$\begin{aligned} &f(0) = g \\ &f(Sy) = h(y, f(y)) \end{aligned}$$

with the corresponding program

```
$func := g$
For $y = 0$ to $n - 1$
    $func := h(y, func)$
Loop
```

So long as h is already computable, the value of $f(n)$ will be computable by the use of a 'for' loop that terminates with the required value in the register *func*.

Similarly, of course, for many-place functions. For example, the value of the two-place function $f(m, n)$ is calculated by

```
$func := g(m)$
For $y = 0$ to $n - 1$
    $func := h(m, y, func)$
Loop
```

which again fixes a value so long as g and h are computable.

Now, our mini-program for the factorial calls the multiplication function which can itself be computed by a similar 'for' loop (invoking addition). And addition can in turn be computed by another 'for' loop (invoking the successor). So reflecting the downward chain of recursive definitions

factorial ⇒ multiplication ⇒ addition ⇒ successor

there's a program for the factorial containing nested 'for' loops, which ultimately calls the primitive operation of incrementing the contents of a register by one (or other operations like setting a register to zero, corresponding to the zero function, or copying the contents of a register, corresponding to an identity function).

The point obviously generalizes: *primitive recursive functions are computable by a series of (possibly nested) 'for' loops.*

Importantly, the converse point also holds. Take a 'for' loop, which calls on two known p.r. functions g and h (with g being used to fix the initial value(s) of a new function f, and h being used on each loop to fix the next value of f as some key argument is incremented). Then this plainly corresponds to a definition by recursion of f in terms of g and h. So if a function can be computed by a program using just 'for' loops as its main programming structure – with the program's 'built in' functions all being p.r. – then the newly defined function will also be primitive recursive.

This gives us a quick-and-dirty way of convincing ourselves that a new function is p.r.: *sketch out a routine for computing it and check that it can all be done with a succession of (possibly nested) 'for' loops which only invoke already known p.r. functions: then the new function will be primitive recursive.*[6]

9.5 Not all computable numerical functions are p.r.

We have seen that any p.r. function is mechanically computable. *But not all computable numerical functions are primitive recursive.* In this section, we first make the claim that there are computable-but-not-p.r. numerical functions look plausible. Then we'll actually cook up an example.[7]

First, then, some plausibility considerations. We've seen that a primitive recursive function f can be computed by a program involving 'for' loops as its main programming structure. Each loop goes through a specified number of iterations. So, just by examining the program for f, we can derive a function s_f, where $s_f(n)$ gives the number of steps it takes to compute $f(n)$. Moreover, to put it crudely, s_f will be definable in terms of repeated additions and multiplications corresponding to the way that 'for' loops are chained together and/or embedded inside each other in the program for f: so s_f will itself be a p.r. function. In sum, *the length of the computation of a p.r. function is given by a p.r. function.*

However, back in Section 2.1 we allowed procedures to count as computational even when don't have nice upper bounds on the number of steps involved. In particular, we allowed computations to involve open-ended searches, with no

[6] We can put all that a bit more carefully. Imagine a simple programming language LOOP. A particular LOOP program operates on a finite set of registers. At the most basic level, the language has instructions for setting the contents of a register to zero, copying contents from one register to another, and incrementing the contents of a register by one. And the *only* important programming structure is the 'for' loop. Such a loop involves setting a register with some initial contents (at the zero-th stage of the loop) and then iterating a LOOP-defined process n times (where on each loop, the process is applied to the result of its own previous application), which has just the effect of a definition by recursion. Such loops can be nested. And sets of nested LOOP commands can be concatenated so that e.g. a loop for evaluating a function g is followed by a loop for evaluating h: concatenation evidently corresponds to composition of functions. Even without going into any more details, it is very easy to see that every LOOP program will indeed define a p.r. function, and every p.r. function is defined by a LOOP program. For a proper specification of LOOP and proofs see Tourlakis (2002); the idea of such programs goes back to Meyer and Ritchie (1967).

[7] Probably no one will regard our cooked-up example as one that might be encountered in ordinary mathematical practice: in fact, it requires a bit of ingenuity to come up with a 'natural' example, though we'll give one later, in the Section **??** where we introduce so-called Ackermann functions.

prior bound on the length of search. We made essential use of this permission in Section 3.6, when we showed that negation complete theories are decidable – for we allowed the process 'enumerate the theorems and wait to see which of φ or $\neg\varphi$ turns up' to count as a computational decision procedure.

And standard computer languages of course have programming structures which implement just this kind of unbounded search. Because as well as 'for' loops, they allow 'do until' loops (or equivalently, 'do while' loops). In other words, they allow some process to be iterated until a given condition is satisfied – *where no prior limit, and so in particular no p.r. limit, is put on the the number of iterations to be executed.*

If we count what are presented as unbounded searches as computations, then it looks very plausible that not everything computable will be primitive recursive.

However, that is as yet only a plausibility consideration: for all we've so far strictly established, it might still be the case that computations presented as unbounded searches can always somehow be turned into procedures with a p.r. limit on the number of steps. But in fact that's false:

> **Theorem 12** *There are algorithmically computable numerical functions which aren't primitive recursive.*

Proof sketch The set of p.r. functions is effectively enumerable. That is to say, we can mechanically produce a list of functions f_0, f_1, f_2, ..., such that each of the f_i is p.r., and each p.r. function appears somewhere on the list.

This holds because, by definition, every p.r. function has a 'recipe' in which it is defined by recursion or composition from other functions which are defined by recursion or composition from other functions which are defined ... ultimately in terms of some primitive starter functions. So choose some standard formal specification language for representing these recipes. Then we can effectively generate 'in alphabetical order' all possible strings of symbols from this language; and as we go, we select the strings that obey the rules for being a recipe for a p.r. function (that's a mechanical procedure). That generates a list of recipes which effectively enumerates the p.r. functions, repetitions allowed.

	0	1	2	3	...
f_0	$\underline{f_0(0)}$	$f_0(1)$	$f_0(2)$	$f_0(3)$	...
f_1	$f_1(0)$	$\underline{f_1(1)}$	$f_1(2)$	$f_1(3)$	...
f_2	$f_2(0)$	$f_2(1)$	$\underline{f_2(2)}$	$f_2(3)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$\underline{f_3(3)}$	...
...	...	...	...	...	$\searrow$

Now take such an effective enumeration f_0, f_1, f_2, ..., of the p.r functions and construct a corresponding *diagonal* function, defined as $d(n) = f_n(n) + 1$

(cf. Section 2.2, and compare the table above). Down the table we list off the p.r. functions. An individual row then gives the values of the p.r. function f_n for each argument. To compute $d(n)$, we just run our effective enumeration of the p.r. functions until we get to f_n. We evaluate that function for the argument n. We then add one. Each step is an entirely mechanically one. So our diagonal function is algorithmically computable. By construction, however, the function d can't be primitive recursive. For suppose otherwise. Then the function d must appear somewhere in the enumeration of p.r. functions, i.e. be the function f_d for some index number d. But now ask what the value of $d(d)$ is. By hypothesis, the function d is none other than the function f_d, so $d(d) = f_d(d)$. But by the initial definition of the diagonal function, $d(d) = f_d(d) + 1$. Contradiction.

Hence d is a computable function which is not primitive recursive. ⊠

'But hold on! *Why* is the diagonal function not a p.r. function?' Well, as we just noted, if f is a p.r. function, then – as we compute $f(n)$ for increasing values of n – the lengths of the successive computations will be given by the successive values of some function $s_f(n)$, where s_f is also primitive recursive. Now contrast evaluating $d(n)$ for increasing values of n. For each new argument, we will have to evaluate a *different* function f_n for that argument (and then add 1). We have no reason to expect there will be a nice pattern in the lengths of these successive computations of all the different functions f_n. In particular, we have no reason to expect there will be a single p.r. function that gives the length of those different computations. And our diagonal argument in effect shows that there isn't one.

9.6 Defining p.r. properties and relations

The p.r. functions then are a large and important class of computable functions. We now want to extend the idea of primitive recursiveness and introduce the ideas of p.r. (numerical) *properties* and *relations*. These form a large and important class of decidable properties and relations.

Now, we can tie talk of functions and talk of properties/relations together by using the very simple but crucial notion of a *characteristic function*. Here's a definition.

> The *characteristic function* of the numerical property P is the one-place function c_P such that if m is P, then $c_P(m) = 0$, and if m isn't P, then $c_P(m) = 1$.
>
> The characteristic function of the two-place numerical relation R is the two-place function c_R such that if m is R to n, then $c_R(m, n) = 0$, and if m isn't R to n, then $c_R(m, n) = 1$.

And similarly for many-place relations. The choice of values for the characteristic function is, of course, entirely arbitrary: any pair of distinct numbers would do. Our choice is supposed to be reminiscent of the familiar use of 0 and 1, one way round or the other, to stand in for *true* and *false*. And our selection of 0 rather than 1 for *true* is simply for later convenience.

The numerical property P partitions the numbers into two sets, the set of numbers that have the property and the set of numbers that don't. Its corresponding characteristic function c_P also partitions the numbers into two sets, the set of numbers the function maps to the value 0, and the set of numbers the function maps to the value 1. And these are the *same* partition. So in a good sense, P and its characteristic function c_P contain exactly the same information about a partition of the numbers: hence we can move between talk of a property and talk of its characteristic function without loss of information. Similarly, of course, for relations.

In what follows, we'll frequently use this link between properties and relations and their characteristic functions in order to carry over ideas defined for functions and apply them to properties/relations. For example:

1. We can now officially say that a numerical property is *decidable* – i.e. a suitably programmed computer can decide whether the property obtains – just if its characteristic function is *(total and) computable*.[8]

2. And without further ado, we can now introduce the idea of a *p.r. property*, meaning – of course – a property with a p.r. characteristic function, and likewise a *p.r. relation* is a relation with a p.r. characteristic function.

9.7 Some more examples

(a) We'll finish the chapter by giving some more examples of p.r. functions, properties and relations, and then proving that they *are* p.r.

Strictly speaking, you can cheerfully skip all this section since we only pick up its details in other sections that you can also skip. On the other hand, in proving Gödel's Theorems, we will need to claim that some key functions and relations are p.r; and our claims will seem a *lot* more plausible if you have already worked through some simpler cases. It is therefore probably worth at least skimming through this section: but you needn't do more than skim.

Here, then, are our examples – and we choose cases mostly to do with primes and prime factorization because such cases will be of key importance later.

1. Let $sg(n) = 0$ for $n = 0$, and $sg(n) = 1$ otherwise. Then sg is primitive recursive. And let $sg(n) = 1$ for $n = 0$, and $sg(n) = 0$ otherwise. Then sg is also primitive recursive.

2. The relations $m = n$, $m < n$ and $m \leq n$ are primitive recursive.

3. The relation $m|n$ that holds when m is a factor of n is primitive recursive.

[8] Compare Section 2.1. The characteristic function needs to be total because it needs to deliver a verdict about each number as to whether it has the property in question.

4. Let $Prime(n)$ be true just when n is a prime number. Then $Prime$ is a p.r. property.[9]

5. List the primes as $\pi_0, \pi_1, \pi_2, \ldots$. Then the function $\pi(n)$ whose value is π_n is p.r.

6. Let $exp(n, i)$ be the – possibly zero – exponent of the prime number π_i in the factorization of n.Then exp is a p.r. function.

7. Let $len(0) = len(1) = 0$; and when $n > 1$, let $len(n)$ be the 'length' of n's factorization, i.e. the number of distinct prime factors of n. Then len is again a p.r. function.

You might well want to pause here to convince yourself that all these are indeed p.r. by the quick-and-dirty method of sketching out how you compute the relevant (characteristic) functions without doing any open-ended searches, just by using 'for' loops.

(b) We'll now show more carefully that those examples are, as claimed, all primitive recursive. However – like the arguments in Section 7.2 – the mini-proofs that follow don't involve anything deep or illuminating. In giving them, we are just solving a few light-weight brain-teasers. So don't let yourself get bogged down: when you find your enthusiasm for this sort of thing waning, skim on quickly to the next chapter. And note, by the way, that we are doing informal everyday mathematics here (i.e. we aren't producing proofs in a formal system, though for brevity's sake we borrow some formal symbols like the connectives).

Proof for (1) We just note that

$$sg(0) = 0$$
$$sg(Sy) = SZ(sg(y))$$

where $SZ(u)$ is p.r. by composition, and $SZ(sg(y)) = S0 = 1$. We prove $sg(y)$ is p.r. similarly (exercise!). ⊠

Proof for (2) The characteristic function of $m = n$ is $sg(|m - n|)$, where $|m - n|$ is the absolute difference function we showed to be p.r. in Section 9.1. The characteristic functions of $m < n$ and $m \leq n$ are $sg(Sn \dot{-} m)$ and $sg(n \dot{-} m)$ respectively. These are all compositions of p.r. functions, and hence themselves p.r. ⊠

(c) Those were easy warm-ups. Now things get just a little more interesting. We begin by listing three useful general facts that we'll use in our remaining proofs:

A. If $f(\vec{x})$ is an n-place p.r. function, then the corresponding relation expressed by $f(\vec{x}) = y$ is an $n + 1$-place p.r. relation.

[9] Remember the useful convention: capital letters for the names of predicates and relations, small letters for the names of functions.

B. Any truth-functional combination of p.r. properties and relations is p.r.

C. Any property or relation defined from a p.r. property or relation by bounded quantifications is also p.r.

And now suppose we introduce the *minimization* operator 'μx', to be read: 'the least x such that …'. Much later, we'll be considering the general use of this operator, but here we will be concerned with *bounded minimization.* So we write

$$f(n) = (\mu x \leq n)P(x)$$

when f takes the number n as argument and returns as value the least number $x \leq n$ such that $P(x)$ if such an x exists, or returns n otherwise. Then we have a fourth useful fact:

D. If P is a p.r. property, then the function $f(n) = (\mu x \leq n)P(x)$ is p.r. And generalizing, suppose that $g(n)$ is a p.r. function, and P is a p.r. property; then $f'(n) = (\mu x \leq g(n))P(x)$ is also p.r.

On a moment's reflection, these claims (A) to (D) should all look plausible (why?): but we'd better prove them.

Proof for (A) We illustrate with the case where f is a one-place function. The characteristic function of the relation expressed by $f(x) = y$ – i.e. the function $c(x, y)$ whose value is 0 when $f(x) = y$ and is 1 otherwise – is given by

$$c(x, y) = sg(|f(x) - y|)$$

Again, the right-hand side is a composition of p.r. functions. ☒

Proof for (B) Suppose $p(x)$ is the characteristic function of the property P. It follows that $sg(p(x))$ is the characteristic function of the property *not-P*, since sg simply flips the two values 0 and 1. But by simple composition of functions, $sg(p(x))$ is p.r. if $p(x)$ is. Hence if P is a p.r. property, so is *not-P*.

Similarly, suppose that $p(x)$ and $q(x)$ are the characteristic functions of the properties P and Q respectively. $p(n) \times q(n)$ takes the value 0 so long as either n is P or n is Q, and takes the value 1 otherwise. So $p(x) \times q(x)$ is the characteristic function of the disjunctive property of being either P or Q; and by composition, $p(x) \times q(x)$ is p.r. if both $p(x)$ and $q(x)$ are. So the disjunction of p.r. properties is another p.r. property.

But any truth-functional combination of properties is definable in terms of negation and disjunction. Which completes the proof. ☒

Proof for (C) Just reflect that checking to see whether e.g. $(\exists x \leq n)Px$ involves using a 'for' loop to check through the cases from 0 to n to see whether Pn holds. Likewise, if f is p.r., checking to see whether $(\exists x \leq f(n))Px$ involves calculating $f(n)$ and then using a 'for' loop to check through the cases from 0 to $f(n)$ to see whether Pn holds. It follows that, if f is p.r., then so are both of

$$K(n) =_{\text{def}} (\exists x \leq n)Px$$
$$K'(n) =_{\text{def}} (\exists x \leq f(n))Px$$

Putting that more carefully, suppose that $p(x)$ is P's p.r. characteristic function. And by composition define the p.r. function $h(u, v) = (p(Su) \times v)$. We put

$$k(0) = p(0)$$
$$k(Sy) = h(y, k(y))$$

so we have

$$k(n) = p(n) \times p(n-1) \times \ldots \times p(1) \times p(0)$$

Then k is K's characteristic function – i.e. the function such that $k(n) = 1$ until we get to an n such that n is P, and then $k(n)$ goes to zero, and thereafter stays zero. Since k is p.r., K is p.r. by definition.

And to get the generalized result, we just note that $K'(n) = K(f(n))$ so is p.r. by composition. We also have similar results for bounded universal quantifiers; we can apply the bounded quantifiers to relations as well as monadic properties; and the bounded quantifiers can equally use '$<$' rather than '$\leq$'. ☒

Proof for (D) Again suppose p is the characteristic function of P, and define k as in the last proof. Then consider the function defined by

$$f(0) = 0$$
$$f(n) = k(n-1) + k(n-2) + \ldots + k(1) + k(0), \text{ for } n > 0$$

Now, $k(j) = 1$ for each j that isn't P, and $k(j)$ goes to zero and stays zero as soon as soon as we hit a j that is P. So $f(n) = (\mu x \leq n)P(x)$, i.e. $f(n)$ returns either the least number that is P, or n, whichever is smaller. So we just need to show that f so defined is indeed primitive recursive. Well, use composition to define the p.r. function $h'(u, v) = (k(u) + v)$, and then put

$$f(0) = 0$$
$$f(Sy) = h'(y, f(y))$$

Which proves the first, simpler, part of Fact D. For the generalization, just note that by the same argument we have $f(g(n)) = (\mu x < g(n))P(x)$ is p.r. if g is, so we can put $f'(n) = f(g(n))$ and we are done. ☒

(d) Given those general facts (A) to (D), we can now prove that the five remaining claims about various properties and functions to do with primes and factorization being primitive recursive:

Proof for (3) We have

$$m|n \leftrightarrow (\exists y \leq n)(0 < y \ \wedge\ 0 < m \ \wedge\ m \times y = n)$$

The relation expressed by the subformula after the quantifier is a truth-functional combination of p.r. relations (multiplication is p.r., so the last conjunct is p.r. by Fact A). So that relation is p.r. by Fact B. Hence $m|n$ is a p.r. relation by Fact C. ☒

Proof for (4) The property of being *Prime* is p.r. because

$$\begin{aligned} \mathit{Prime}(n) \leftrightarrow n \neq 1 \ \wedge \ (\forall u \leq n)(\forall v \leq n)(u \times v = n \\ \rightarrow \ (u = 1 \vee v = 1)) \end{aligned}$$

and the r.h.s is built up from p.r. components by truth-functional combination and restricted quantifiers. (Here we rely on the trivial fact that the factors of n cannot be greater than n.) ☒

Proof for (5) The function π_n, whose value is the n-th prime (counting from zero), is p.r. – for consider the definition

$$\begin{aligned} &\pi_0 = 2 \\ &\pi_{Sn} = (\mu x \leq n! + 1)(\pi_n < x \wedge \mathit{Prime}(x)) \end{aligned}$$

where we rely on the familiar fact that the next prime after n is no greater than $n! + 1$ and use the generalized version of Fact D. ☒

Proof for (6) This function is well-defined because by the so-called Fundamental Theorem of Arithmetic, which says that numbers have a unique factorization into primes.

No exponent in the prime factorization of n is larger than n itself, so we have

$$\mathit{exp}(n, i) = (\mu x \leq n)\{(\pi_i^x | n) \ \wedge \ \neg(\pi_i^{x+1} | n)\}$$

That is to say, the desired exponent of π_i is the number x such that π_i^x divides n but π_i^{x+1} doesn't: note that $\mathit{exp}(n, k) = 0$ when π_k isn't a factor of n. Again, our definition of *exp* is built out of p.r. components by operations that yield another p.r. function. ☒

Proof for (7) $(\mathit{Prime}(m) \wedge m|n)$ holds when m is a prime factor of n. This a p.r. relation (being a conjunction of p.r. properties/relations). So it has a p.r. characteristic function we'll abbreviate $\mathit{pf}(m, n)$. Now consider the function

$$p(m, n) = \mathit{sg}(\mathit{pf}(m, n))$$

Then $p(m, n) = 1$ just when m is a prime factor of n and is zero otherwise. So

$$\mathit{len}(n) = p(0, n) + p(1, n) + \ldots + p(n - 1, n) + p(n, n)$$

So to give a p.r. definition of *len*, we can first put

$$\begin{aligned} &l(x, 0) = p(0, x) \\ &l(x, Sy) = (p(Sy, x) + l(x, y)) \end{aligned}$$

And then finally put $len(n) = l(n, n)$. ⊠

Well, all good clean fun if you like that kind of thing. But as I said before, don't worry if you don't! For having shown that these kinds of results *can* be proved, you can now very cheerfully forget the details of how to do it.

10 Capturing functions

In this chapter we work up to the important idea of a *p.r. adequate theory* of arithmetic, i.e. one that can capture all p.r. functions, properties and relations. Then, in the next chapter, we will show that Q and hence PA is p.r. adequate.

However, we haven't yet explained what is involved in capturing a *function* as opposed to a property or relation, and there is a slight wrinkle here. We therefore need to spend a little time explaining that idea, which is (I'm afraid) rather boring housekeeping.

10.1 Expressing and capturing functions

Suppose f is a one-place (total) numerical function. And suppose m has the relation R_f to n just in case $f(m) = n$. Then we'll say R_f is f's *corresponding relation*. Functions and their corresponding relations match up pairs of things in exactly the same way: so f and R_f have exactly the same extension, namely the set of ordered pairs $\langle m, f(m)\rangle$.[1]

And just as the characteristic function trick (Section 9.6) allows us to take ideas defined for functions and apply them to properties and relations, *this* very simple tie between functions and their corresponding relations allows us to carry over ideas defined for relations and apply them to functions.

For a start, consider how we can use this tie to extend the idea of *expressing* a relation to cover the idea of expressing a function using an open wff. Here again is the familiar definition for relations, now applied to R_f:

> A two-place numerical relation R_f is expressed by $\varphi(\mathsf{x}, \mathsf{y})$ in an (interpreted) arithmetical language L just if, for any m, n,
>
> if m has the relation R_f to n, then $\varphi(\mathsf{m}, \mathsf{n})$ is true,
>
> if m doesn't have relation R_f to n, then $\neg\varphi(\mathsf{m}, \mathsf{n})$ is true.

Moving from the relation R_f to the function f, this naturally becomes:

> A one-place numerical function f is expressed by $\varphi(\mathsf{x}, \mathsf{y})$ in an (interpreted) arithmetical language L just if, for any m, n,
>
> if $f(m) = n$, then $\varphi(\mathsf{m}, \mathsf{n})$ is true,
>
> if $f(m) \neq n$, then $\neg\varphi(\mathsf{m}, \mathsf{n})$ is true.

The generalization to many-place functions is immediate.

[1] For that reason, many logicians would simply *identify* a function and its corresponding relation. We won't pause to argue the pros and cons of taking that line.

Similarly, we can extend the idea of *capturing* from relations to functions. Here is the definition again for a two-place relation R_f:

> A two-place numerical relation R_f is captured by $\varphi(\mathsf{x},\mathsf{y})$ in theory T just if, for any m, n,
>
> if m has the relation R_f to n, then $T \vdash \varphi(\mathsf{m},\mathsf{n})$,
>
> if m does not have the relation R_f to n, then $T \vdash \neg\varphi(\mathsf{m},\mathsf{n})$.

And we can naturally say that a one-place function f (i.e., by abuse of notation, the function $f(x) = y$) is captured by $\varphi(\mathsf{x},\mathsf{y})$ in theory T so long as that wff captures the corresponding relation R_f. Which comes to the following:

> A one-place numerical function f is captured by $\varphi(\mathsf{x},\mathsf{y})$ just if, for any m, n,
>
> if $f(m) = n$, then $T \vdash \varphi(\mathsf{m},\mathsf{n})$,
>
> if $f(m) \neq n$, then $T \vdash \neg\varphi(\mathsf{m},\mathsf{n})$.

Again, the generalization to many-place functions is immediate.

10.2 'Capturing as a function'

So far so good. However, although our definition above is the natural analogue of our definition of what it takes to capture a relation, it is convenient and nowadays standard to work with a stronger notion of capturing a function. This section explains the stronger notion.

Our previous definition might be said to be weak in the following sense. It tells us that T captures a function f if there is some φ which captures the relation that holds between m and n when $f(m) = n$. But it doesn't require that φ – so to speak – captures the function *as a function*, i.e. it doesn't require that T can prove that the capturing wff φ relates a given m to exactly one value n. We will now impose this extra requirement, and say:

> The one-place function f is *captured as a function* by $\varphi(\mathsf{x},\mathsf{y})$ in theory T just if
>
> (i) for every m, $T \vdash \exists!\mathsf{y}\varphi(\mathsf{m},\mathsf{y})$
>
> and for any m, n:
>
> (ii) if $f(m) = n$ then $T \vdash \varphi(\mathsf{m},\mathsf{n})$,
>
> (iii) if $f(m) \neq n$, then $T \vdash \neg\varphi(\mathsf{m},\mathsf{n})$.

Here '$\exists!\mathsf{u}$' is the standard uniqueness quantifier, to be read 'there is exactly one u such that ...'.[2] So the new clause (i), as we want, insists that the putative capturing relation can be proved to relate each numerical argument to some unique value: in a phrase, the relation is (provably) functional. Again, the generalization to many-place functions is immediate.

[2] Let '$\exists!$' be defined by taking '$\exists!\mathsf{u}\varphi(\mathsf{u})$' as short for '$\exists\mathsf{u}(\varphi(\mathsf{u}) \land \forall\mathsf{v}(\varphi(\mathsf{v}) \to \mathsf{v} = \mathsf{u}))$'. We won't fuss about how to handle any potential clash of variables.

Note however that, even for very modest theories like Q, conditions (i) and (ii) in fact imply (iii).

Proof Suppose $f(m) \neq n$ because $f(m) = k$, where $n \neq k$. Suppose further that (i) and (ii) hold, so $\varphi(\mathsf{x}, \mathsf{y})$ is provably functional, and also $T \vdash \varphi(\mathsf{m}, \mathsf{k})$. Then by simple logic, (i) and (ii) imply that $T \vdash \mathsf{n} \neq \mathsf{k} \rightarrow \neg\varphi(\mathsf{m}, \mathsf{n})$. But as we saw in Section 6.1, even when T is mere Baby Arithmetic, if $n \neq k$, then $T \vdash \mathsf{n} \neq \mathsf{k}$. Hence, if T contains Baby Arithmetic, (iii) if $f(m) \neq n$ then $T \vdash \neg\varphi(\mathsf{m}, \mathsf{n})$. ⊠

Therefore, to confirm in particular that φ captures f as a function in Q or any stronger theory, *we only need to check that conditions (i) and (ii) hold.* That is why capturing-as-a-function is very often defined just in terms of (i) and (ii) holding.

And to link up with a third common definition also found in the literature, assume that T contains Baby Arithmetic and is consistent. Then the definition in terms of conditions (i) and (ii) is easily seen to be equivalent to this:

> The one-place function f is captured as a function by the $\varphi(\mathsf{x}, \mathsf{y})$ in theory T just if for any m, n:
> if $f(m) = n$, then $T \vdash \forall \mathsf{y}(\varphi(\mathsf{m}, \mathsf{y}) \leftrightarrow \mathsf{y} = \mathsf{n})$.

Likewise, of course, for many-place functions.

10.3 'If capturable, then capturable as a function'

Trivially, if φ captures f as a function in T, then φ captures f in the weaker sense of Section 10.1.

The strict converse doesn't obtain. However, we have a result which is almost as good. For suppose T is either Q or extends Q, so T proves everything that Q does: then if φ captures the function f in T, then there will always be a closely related wff $\widetilde{\varphi}$ which *does* capture f *as a function* in T.

Let's illustrate our claim for the case of a one-place function. So suppose φ captures f in T. And now consider the wff $\widetilde{\varphi}$ defined as follows:

$$\widetilde{\varphi}(\mathsf{x}, \mathsf{y}) =_{\text{def}} \varphi(\mathsf{x}, \mathsf{y}) \wedge (\forall \mathsf{z} \leq \mathsf{y})(\varphi(\mathsf{x}, \mathsf{z}) \rightarrow \mathsf{z} = \mathsf{y})$$

Then, for a given m, $\widetilde{\varphi}(\mathsf{m}, \mathsf{x})$ is satisfied by a *unique* n, i.e. the smallest n such that $\varphi(\mathsf{m}, \mathsf{n})$ is true. It is easy to show that this wff not only also captures f but captures it as a function (so long as T is at least as strong as Q). Why? Essentially because, as we know from Section 7.2, Q is good at proving results involving bounded quantifiers. In detail (just for enthusiasts!):

Proof Assume we are dealing with a theory T which proves everything Q proves. Suppose that φ captures in T the one-place function f. We need to show

i. for every m, $T \vdash \exists! \mathsf{y} \widetilde{\varphi}(\mathsf{m}, \mathsf{y})$,
ii. if $f(m) = n$ then $T \vdash \widetilde{\varphi}(\mathsf{m}, \mathsf{n})$.

So suppose $f(m) = n$. Since the value of $f(m)$ is unique, that means $f(m) \neq k$ for all $k < n$. Because φ captures f in T, that means (a) $T \vdash \varphi(\mathsf{m}, \mathsf{n})$, and (b) for $k < n$, $T \vdash \neg\varphi(\mathsf{m}, \mathsf{k})$. But (b) implies (c): for $k \leq n$, $T \vdash \varphi(\mathsf{m}, \mathsf{k}) \rightarrow \mathsf{k} = \mathsf{n}$. And from (c) and Result 4 of Section 7.2, we get (d) $T \vdash (\forall \mathsf{x} \leq \mathsf{n})(\varphi(\mathsf{m}, \mathsf{x}) \rightarrow \mathsf{x} = \mathsf{n})$. Putting (a) and (d) together, that means $T \vdash \widetilde{\varphi}(\mathsf{m}, \mathsf{n})$, which establishes (ii).

Since $T \vdash \widetilde{\varphi}(\mathsf{m}, \mathsf{n})$, to establish (i) it is now enough to show that, for arbitrary a, $T \vdash \widetilde{\varphi}(\mathsf{m}, \mathsf{a}) \rightarrow \mathsf{a} = \mathsf{n}$. So, arguing in T, suppose $\widetilde{\varphi}(\mathsf{m}, \mathsf{a})$, i.e. $\varphi(\mathsf{m}, \mathsf{a}) \wedge (\forall \mathsf{z} \leq \mathsf{a})(\varphi(\mathsf{m}, \mathsf{z}) \rightarrow \mathsf{z} = \mathsf{a})$. By Result 8 of Section 7.2, $\mathsf{a} \leq \mathsf{n} \vee \mathsf{n} \leq \mathsf{a}$. If the first, (d) yields $\varphi(\mathsf{m}, \mathsf{a}) \rightarrow \mathsf{a} = \mathsf{n}$, and so $\mathsf{a} = \mathsf{n}$. If the second, then $\varphi(\mathsf{m}, \mathsf{n}) \rightarrow \mathsf{n} = \mathsf{a}$, so $\mathsf{n} = \mathsf{a}$. So either way $\mathsf{a} = \mathsf{n}$. Discharge the supposition, and we're done. ⊠

The result generalizes, of course, to the case where we are dealing with a wff $\varphi(\vec{\mathsf{x}}, \mathsf{y})$ which captures a many-place function $f(\vec{x})$. Just define the corresponding $\widetilde{\varphi}(\vec{\mathsf{x}}, \mathsf{y})$ in the analogous way (replacing 'x' by '$\vec{\mathsf{x}}$'), and $\widetilde{\varphi}$ will capture f as a function.

In sum – once we are dealing with arithmetics as strong as Q – if a function is capturable at all it is capturable-as-a-function. Which is, of course, why many treatments only bother to introduce the second notion.[3]

10.4 Capturing functions, capturing properties

Note that our various definitions hang together in the following rather convenient way. Suppose T contains quantifiers, and (like Q) can prove that $0 \neq 1$; then *a property is capturable by T if and only if its characteristic function is capturable as a function*:

Proof Suppose P is captured in T by the open wff $\varphi(\mathsf{x})$, and consider the wff

$$((\varphi(\mathsf{x}) \leftrightarrow \mathsf{y} = 0) \wedge (\neg\varphi(\mathsf{x}) \leftrightarrow \mathsf{y} = 1))$$

It is easily seen that this two-place relational expressions captures c_P, the characteristic function of P, and captures it as a function. Conversely, suppose the wff $\varphi(\mathsf{x}, \mathsf{y})$ captures the characteristic function c_P; then the wff $\varphi(\mathsf{x}, 0)$ captures the corresponding property P. ⊠

So instead of laying down separate conditions for properties/relations and functions being capturable, we could have initially just given conditions for the case of functions, and then let properties and relations look after themselves by saying that they are capturable if their characteristic functions are.

[3] I have laboured a bit over these variant definitions in part to help comparison with the ideas in other books. Again terminology varies widely. For the pair of ideas 'capturing a function' and 'capturing a function as a function' we find e.g. 'weakly defines'/'strongly defines' (Smullyan, 1992, p. 99), 'defines'/'represents' (Boolos et al., 2002, p. 207), 'represents'/'functionally represents' (Cooper, 2004, pp. 56, 59). While those who only highlight the idea of capturing-as-a-function sometimes use e.g. 'defines' for *that* notion (Lindström, 2003, p. 9), though plain 'represents' seems most common (Mendelson, 1997, p. 171), (Epstein and Carnielli, 2000, p. 192).

10.5 The idea of p.r. adequacy

(a) Our three formal arithmetics BA, Q, and PA, have function symbols for successor, addition and multiplication built in. Plainly BA can't capture those three functions as functions (for that requires the use of wffs with quantifiers, which BA lacks) – though it does capture those functions in the weaker sense. For example $\mathsf{S}\xi = \zeta$ captures the successor function.[4] However, in Q (and so PA) successor, addition and multiplication *are* captured as functions. To take just one example, $\mathsf{x} + \mathsf{y} = \mathsf{z}$ captures addition as a function in Q. To establish this, we need to confirm that

i. for every m, n, $\mathsf{Q} \vdash \exists!\mathsf{z}(\mathsf{m} + \mathsf{n} = \mathsf{z})$

which is trivial by the logic of identity. We also need to confirm that for any m, n, o,

ii. if $m + n = o$, then $\mathsf{Q} \vdash \mathsf{m} + \mathsf{n} = \mathsf{o}$

But we've already seen that Q can prove any true (unquantified) equation.

(b) Of course, we want any reasonably competent formal arithmetic to be able to deal with more than addition, multiplication and successor. Recall, the value of a p.r. function for any given argument(s) is computable – in p.r. bounded time – in accordance with a step-by-step algorithm. But, as we've said before, the whole aim of formalization is to systematize and regiment what we can already do. And if we can informally calculate the value of a p.r. function for a given input in an entirely mechanical way – ultimately by just repeating lots of school-arithmetic operations – then we will surely want to aim for a formal arithmetic which is able to track these informal calculations. So it seems that we will want a formal arithmetic worth its keep to be able to express any p.r. function and prove, case-by-case, the correct results about the function's values for specific arguments.[5] That motivates a pair of definitions:

> A theory T is *weakly p.r. adequate* if, for every p.r. function f, there is a corresponding φ in T that captures it.
>
> A theory T is *p.r. adequate* if for every p.r. function f, there is a corresponding φ in T that captures it as a function.

(c) Now, there's an easy, brute-force, way of constructing a weakly p.r. adequate theory. Start again from BA, our theory of Baby Arithmetic (see Section 6.1). This, recall, is a quantifier free theory which has schemas which reflect

[4] Here we take up the permission we gave ourselves in Section 4.4, fn. 4 to read the variables in the official definitions of expressing/capturing as serving as placeholders when necessary.

[5] Compare the informal idea of being 'sufficiently strong' that we met in Section 5.1. The informal idea was about capturing any decidable property, i.e. any property with a *computable* characteristic function: while being p.r. adequate is a matter of capturing *primitive recursive* functions. And we know that there are computable functions which aren't p.r. So, at least on the face of it, the informal idea is stronger.

the p.r. definitions of addition and multiplication. As we showed, we can use instances of these schemas to prove any true equation or inequation using successor, addition and multiplication. Hence BA is adequate for those three functions in the sense that it can evaluate them correctly case-by-case for specific arguments.

So far, so good. And next suppose we start expanding BA by adding new vocabulary and new schemas. As a first step, we can add the symbol '$\uparrow$', intended to express the exponential function, and then say that all numeral instances of the following are axioms too:

Schema 7 $\zeta \uparrow \mathsf{0} = \mathsf{1}$

Schema 8 $\zeta \uparrow \mathsf{S}\xi = (\zeta \uparrow \xi) \times \zeta$

Instances of those schemas enable us to prove the correct result for the value of the exponential function for any arguments. So that makes four functions which can be captured in our expanded BA.

For tidiness, let's resymbolize these using the function symbols '$\mathsf{f_0}$', '$\mathsf{f_1}$', '$\mathsf{f_2}$', '$\mathsf{f_3}$'. And now let's keep going: we will add a symbol '$\mathsf{f_n}$' for each n, with the plan that '$\mathsf{f_n}$' should express the n-th p.r. function f_n in a 'good' effective enumeration of the recipes for p.r. functions (where an enumeration is 'good' if the p.r. definition of f_n only involves functions earlier in the enumeration). Then for each '$\mathsf{f_n}$', we write down schemas involving that function expression which reflect f_n's definition in terms of earlier functions. Call the resulting theory $\mathsf{PRA_0}$.[6]

$\mathsf{PRA_0}$ is still a properly axiomatized theory, because it will be effectively decidable whether any given wff is an instance of one of axiom schemas. Plainly, its language is much richer than BA's, since it has a separate function expression for each primitive recursive function: but for all that, its language remains impoverished in other ways – for it still can't express any general claims. Because it is quantifier-free, we can show that $\mathsf{PRA_0}$ is a negation-complete theory like BA (in fact we just generalize the argument we used to show BA can either prove or disprove every sentence in its limited language). And by construction, $\mathsf{PRA_0}$ can capture all p.r. functions[7] – though, lacking quantifiers, it of course can't be p.r. adequate in the stronger sense.

(d) In sum, we can readily construct a (weakly) p.r. adequate arithmetic by the high-cost method of infinitely expanding the vocabulary of arithmetic and throwing in axioms for every p.r. function. But do we actually *need* to do this?

We don't. In fact, *we only need the language of basic arithmetic in order to frame a (strongly) p.r. adequate theory.* To put it very roughly, the ground we lose by restricting ourselves to a language with successor, addition, and multiplication as the only built-in functions, we can make up again by having quantification available for definitional work. Indeed, even the induction-free arithmetic Q is p.r. adequate. Proving that is work for the next chapter.

[6] That's short for 'quantifier-free Primitive Recursive Arithmetic'. Full PRA is (roughly) $\mathsf{PRA_0}$ plus a weak induction schema.

[7] Footnote 4 above applies again!

11 Q is p.r. adequate

We are going to show that *any* p.r. function – and hence (via its characteristic function) *any* p.r. property and relation – can be captured in Q. In a phrase, Q is p.r. adequate. And it will later turn out that this result takes us most of the way to showing that Q is 'sufficiently strong'.

This chapter is unavoidably proof-packed. So here's a local road-map of the line of argument. Recall the idea of a Σ_1 wff which was introduced in Section 7.3. A Σ_1 wff is the existential quantification of a 'simple' kernel wff (where being 'simple' or Δ_0 means lacking unbounded quantifications). Define a Σ_1 function as one that can be expressed by a Σ_1 wff. Then we can prove two Big Results:

1. *Every Σ_1 function can be captured as a function in* Q. We show this in the first section below, building on Theorem 9, which told us that Q is Σ_1-complete.

2. *Every p.r. function is a Σ_1 function.* This takes us the next four sections to establish. (i) We first use the so-called 'β-function' trick which Gödel invented to prove that L_A has the resources to express any p.r. function. Then (ii) we look at the details of our proof to extract the more detailed information that a Σ_1 wff is always enough to do the expressive job.

Putting those two Big Results together immediately gives us the target result that Q is p.r. adequate. It trivially follows that PA is p.r. adequate too.

Perhaps we should note, though, that the new proof ideas which we need in this chapter to establish the two Big Results are not used again in this book. It is therefore not necessary to master all the fine details of the proofs in order to grasp what follows in later chapters.

11.1 Q can capture all Σ_1 functions

(a) We start with a trio of definitions – definitions which for the moment are being applied to *total* numerical functions:

f is a Δ_0 function if it can be expressed by a Δ_0 wff.
f is a Σ_1 function if it can be expressed by a Σ_1 wff.
f is a Π_1 function if it can be expressed by a Π_1 wff.

Since a Σ_1 wff is, by definition, always equivalent to some strictly Σ_1 wff, it is trivial that for any Σ_1 function there's a *strictly* Σ_1 wff which expresses it – a point we'll later use repeatedly.

Note the 'can' in our definitions. A function f might be expressible by some other kinds of wff too, but it is Π_1 (for example) so long as it *can* also be expressed by some Π_1 wff. Here's a little result that illustrates the point.

The Σ_1/Π_1 *lemma.* If a function is Σ_1 it is also Π_1.

Proof Suppose the one-place function f can be expressed by the strictly Σ_1 wff $\varphi(\mathsf{x},\mathsf{y})$. Since f *is* a function, and maps numbers of unique values, we have $f(m) = n$ if and only if $\forall z(f(m) = z \to z = n)$. Hence $f(m) = n$ if and only if $\forall \mathsf{z}(\varphi(\mathsf{m},\mathsf{z}) \to \mathsf{z} = \mathsf{n})$ is true.[1] In other words, f is equally well expressed by $\forall \mathsf{z}(\varphi(\mathsf{x},\mathsf{z}) \to \mathsf{z} = \mathsf{y})$. But it is a trivial exercise of moving quantifiers around to show that if $\varphi(x, y)$ is strictly Σ_1, then $\forall \mathsf{z}(\varphi(\mathsf{x},\mathsf{z}) \to \mathsf{z} = \mathsf{y})$ is Π_1. ⊠

(b) In Section 7.4, we showed that Q can correctly decide every Δ_0 sentence – i.e. prove it if it is true, refute it if it is false. We've also shown in Section 10.3 that if Q captures a function by some wff φ, it can capture it as a function by a corresponding wff $\widetilde{\varphi}$. And these facts entail

The Q/Δ *lemma.* Q can capture any Δ_0 function as a function.

Proof Suppose the one-place function f is expressed by the Δ_0 wff $\varphi(\mathsf{x},\mathsf{y})$. Then by definition of 'expressing', if $f(m) = n$, then $\varphi(\mathsf{m},\mathsf{n})$ is true, and hence – since Q correctly settles every Δ_0 wff, $\mathsf{Q} \vdash \varphi(\mathsf{m},\mathsf{n})$. Likewise, if $f(m) \neq n$, then $\varphi(\mathsf{m},\mathsf{n})$ is false, and hence $\mathsf{Q} \vdash \neg\varphi(\mathsf{m},\mathsf{n})$. So $\varphi(\mathsf{x},\mathsf{y})$ not only expresses but captures f in Q. Hence $\widetilde{\varphi}(\mathsf{x},\mathsf{y})$ captures f as a function in Q. But it is easy to check that, by the construction of $\widetilde{\varphi}$, this wff is still Δ_0 if φ is. (The argument for many-place functions is, of course, exactly parallel.) ⊠

(c) The rest of this section beefs up that last very easy lemma by using a really delightful bit of sheer ingenuity to establish

Theorem 13 Q *can capture any* Σ_1 *function as a function.*

What does this mean? It means that once we've found a Σ_1 wff which *expresses* a function f, then we know that there is *some* wff or other which *captures* f as a function (it will in fact still be a Σ_1 wff but needn't be the same one).

Our proof falls into two stages. First (the ingenious stage), we show that a Σ_1 function is equivalent to a composition of Δ_0 functions. And then second (the straightforward stage), we show that Q can capture any composition of Δ_0 functions.

Proof: first stage Take a total one-place function f expressed by a strictly Σ_1 open wff with two free variables. Suppose this wff is of the form $\exists \mathsf{z}\mathsf{R}(\mathsf{x},\mathsf{y},\mathsf{z})$ with just one quantifier, where $\mathsf{R}(\mathsf{x},\mathsf{y},\mathsf{z})$ is Δ_0. (The case where the Δ_0 kernel of the wff is preceded by more than one existential quantifier can be dealt with very similarly.)

[1] To avoid clash of variables, assume 'z' doesn't appear in φ.

R will, of course, express some three-place relation R, where $\mathsf{R(m,n,o)}$ is true just when $Rmno$. And then $f(m) = n$ just when $\exists zRmnz$.

Now for the clever trickery![2] First we define a couple more functions:

> $g(x)$ is the least y such that $(\exists u \leq y)(\exists v \leq y)Rxuv$;
> $h(x,y)$ is the least $z \leq y$ such that $(\exists v \leq y)Rxzv$ if such an z exists, or is 0 otherwise.

Since f is total, for every x there are values u, v such that $Rxuv$, and so g is well-defined. And then what's going on here is that $g(m)$ puts a ceiling c on the numbers we need to search through before finding a pair n, o such that $Rmno$ is true. And $h(m,c)$ looks for a number n under the ceiling c such that for some o also under that ceiling $Rmno$ is true. Which means, of course, that

$$f(m) = h(m, g(m))$$

And the point about this redefinition of f as a composition of functions is that both g and h are Δ_0 functions.

Why so? Because our two functions are expressed by, respectively,

$$\begin{aligned}\mathsf{G(x,y)} =_{\text{def}} {} & \mathsf{(\exists u \leq y)(\exists v \leq y)R(x,u,v)} \\ & \wedge \mathsf{(\forall w \leq y)[w \neq y \to \neg(\exists u \leq w)(\exists v \leq w)R(x,u,v)]} \\ \mathsf{H(x,y,z)} =_{\text{def}} {} & \mathsf{[(\exists v \leq y)R(x,z,v) \wedge \neg(\exists u \leq z)(\exists v \leq y)(u \neq z \wedge R(x,u,v))]} \\ & \vee \mathsf{z = 0}\end{aligned}$$

and those wffs are evidently Δ_0.

Proof: second stage So we've shown that a one-place Σ_1 function can be treated as a composition of two Δ_0 functions; and the generalization to many-place functions is straightforward. Now we show that Q not only captures any Δ_0 function (as we've already seen) but also any composition of two Δ_0 functions.

We'll take the simplest case (again, generalizing the argument is easy). So suppose that f is a one-place Σ_1 function, and suppose

$$f(m) = h(m, g(m))$$

where g and h are Δ_0 functions as in the first stage of the proof. So g and h are captured as functions by Δ_0 wffs which we'll abbreviate $\widetilde{\mathsf{G}}\mathsf{(x,y)}$ and $\widetilde{\mathsf{H}}\mathsf{(x,y,z)}$ respectively. Then we'll show that the function $f(x) = h(x, g(x))$ is not only expressed but captured by the Σ_1 wff $\mathsf{F(x,y)} =_{\text{def}} \mathsf{\exists z(\widetilde{G}(x,z) \;\wedge\; \widetilde{H}(x,z,y))}$.

For suppose that $f(m) = n$. Then for some o, $g(m) = o$, and $h(m,o) = n$. Then by the capturing assumption, the following are provable in Q:

$$\mathsf{\forall y(\widetilde{G}(m,y) \leftrightarrow y = o)}$$
$$\mathsf{\forall z(\widetilde{H}(m,o,z) \leftrightarrow z = n)}$$

But by elementary logic, those two imply

[2] Credit where credit is due: I first learnt this neat dodge from (Boolos et al., 2002, p. 206).

$$\forall \mathsf{y}(\exists \mathsf{z}(\widetilde{\mathsf{G}}(\mathsf{m},\mathsf{z}) \wedge \widetilde{\mathsf{H}}(\mathsf{m},\mathsf{z},\mathsf{y})) \leftrightarrow \mathsf{y} = \mathsf{n})$$

But that means we have

If $f(m) = n$, then $\forall \mathsf{y}(\mathsf{F}(\mathsf{m},\mathsf{y}) \leftrightarrow \mathsf{y} = \mathsf{n})$

Which shows that F captures f. ⊠

In sum, then, every Σ_1 functions is a composition of two Δ_0 functions, and such a composition of functions can be captured in Q; so Q can capture as a function every Σ_1 function. Which gives us Theorem 13.

11.2 L_A can express all p.r. functions: starting the proof

Our next main aim – we are at step 2(i), as described in the preamble to this chapter – is to prove

Theorem 14 *Every p.r. function can be expressed in L_A.*

The overall proof strategy Suppose that the following three propositions are all true:

1. L_A can express the initial functions.
2. If L_A can express the functions g and h, then it can also express a function f defined by composition from g and h.
3. If L_A can express the functions g and h, then it can also express a function f defined by primitive recursion from g and h.

Now, any p.r. function f can be specified by a chain of definitions by composition and/or primitive recursion, building up from initial functions. So as we follow through the chain of definitions which specifies f, we start with initial functions which are expressible in L_A, by (1). And – by (2) and (3) – each successive definitional move takes us from expressible functions to expressible functions. So, given (1) to (3) *are* true, f must be expressible in L_A. Hence: in order to prove our theorem, it is enough to to prove (1) to (3).

Proof for (1) This step is trivial. First, the successor function $Sx = y$ is expressed by the open wff $\mathsf{Sx} = \mathsf{y}$. Second, the zero function $Z(x) = 0$ is expressed by the wff $\mathsf{Z}(\mathsf{x},\mathsf{y}) =_{\text{def}} (\mathsf{x} = \mathsf{x} \wedge \mathsf{y} = \mathsf{0})$.

Finally, the three-place identity function $I^3_2(x, y, z) = y$, to take just one example, is expressed by the wff $\mathsf{I}^3_2(\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{u}) =_{\text{def}} (\mathsf{x} = \mathsf{x} \wedge \mathsf{y} = \mathsf{u} \wedge \mathsf{z} = \mathsf{z})$. Likewise for all the other identity functions. (Note, all the initial functions are Δ_0, i.e. are expressible by a Δ_0 wff.) ⊠

Proof for (2) Suppose g and h are one-place functions, expressed by the wffs $\mathsf{G}(\mathsf{x},\mathsf{y})$ and $\mathsf{H}(\mathsf{x},\mathsf{y})$ respectively. Then, the function $f(x) = h(g(x))$ is expressed by the wff $\exists \mathsf{z}(\mathsf{G}(\mathsf{x},\mathsf{z}) \wedge \mathsf{H}(\mathsf{z},\mathsf{y}))$. Other cases where g and/or h are multi-place functions can be handled similarly. ⊠

Starting the proof sketch for (3) Now for the fun part. Consider the primitive recursive definition of the factorial function again:

$$0! = 1$$
$$(Sx)! = Sx \times x!$$

The multiplication and successor functions involved on the right of the second equation here are of course expressible in L_A: but how can we express our defined function in L_A?

Think about the p.r. definition for the factorial in the following way. It tells us how to construct a sequence of numbers $0!, 1!, 2!, \ldots, x!$, where we move from the u-th member of the sequence (counting from zero) to the next by multiplying by Su. Putting $x! = y$, the p.r. definition thus says

A. There is a sequence of numbers $k_0, k_1, \ldots, k_x$ such that: $k_0 = 1$, and if $u < x$ then $k_{Su} = Su \times k_u$, and $k_x = y$.

So the question of how to reflect the p.r. definition of the factorial inside L_A comes to this: how can we express facts about finite sequences of numbers using the limited resources of L_A?

11.3 The idea of a β-function

Let's pause the proof sketch for (3), and think first about the *kind* of trick we could use here.

Suppose $\pi_0, \pi_1, \pi_2, \pi_3, \ldots$ is the series of prime numbers $2, 3, 5, 7, \ldots$. Now consider the number

$$b = \pi_0^{k_0} \cdot \pi_1^{k_1} \cdot \pi_2^{k_2} \cdots \pi_n^{k_n}$$

This single number b can be thought of as encoding the whole sequence $k_0, k_1, k_2, \ldots, k_n$. And we can extract the coded sequence again by using the (primitive recursive) decoding function $exp(b, i)$ which we met in Section 9.7; for this function returns the exponent of the prime number π_i in the factorization of b. By the construction of b, then, $exp(b, i) = k_i$ for $i \leq n$.

Now let's generalize. We'll say

> A two-place *β-function* is a function of the form $\beta(b, i)$ such that, for *any* finite sequence of natural numbers $k_0, k_1, k_2, \ldots, k_n$ there is a code b such that for every $i \leq n$, $\beta(b, i) = k_i$.

So the idea is that – for any finite sequence of numbers you choose – you can select a corresponding code number b to be the first argument for β, and then the function will decode it and spit out the members of the required sequence in order as its second argument is increased.[3]

[3] Referring to such a function as a 'beta-function' is absolutely standard. The terminology was introduced by Gödel himself in his Princeton Lectures (1934, p. 365).

We've just seen that there is nothing in the least bit magical or mysterious about the idea of a β-function: *exp* is a simple example. And evidently, we'll be able to use code numbers and a decoding β-function to talk, in effect, about finite sequences of numbers. However, our first example of a β-function is defined in terms of the exponential function which *isn't* built into L_A.[4] So the obvious next question is: can we construct a β-function just out of successor, addition and multiplication which *are* built into L_A?

It turns out to simplify things if we liberalize our notion of a β-function just a little. So we'll now also consider *three*-place β-functions, which take *two* code numbers c and d, as follows:

> A three-place β-function is a function of the form $\beta(c, d, i)$ such that, for *any* finite sequence of natural numbers $k_0, k_1, k_2, \ldots, k_n$ there is a pair of code numbers c, d such that for every $i \leq n$, $\beta(c, d, i) = k_i$.

A three-place β-function will do just as well as a two-place function to help us express facts about finite sequences.

Even with this liberalization, it still isn't obvious how to define a β-function in terms of the functions built into basic arithmetic. But Gödel neatly solves our problem as follows. Put

$$\beta(c, d, i) =_{\text{def}} \text{the remainder left when } c \text{ is divided by } d(i+1)+1.$$

Then, given any sequence $k_0, k_1, \ldots, k_n$, we can find a suitable pair of numbers c, d such that for $i \leq n$, $\beta(c, d, i) = k_i$.

This claim should look intrinsically plausible. As we divide c by $d(i+1)+1$ for different values of i $(0 \leq i \leq n)$, we'll get a sequence of $n+1$ remainders. Vary c and d, and the sequence of $n+1$ remainders will vary. The permutations as we vary c and d without limit *appear* to be simply endless. We just need to check, then, that appearances don't deceive, and we *can* always find a big enough c and a smaller d which makes the sequence of remainders match a given $n+1$-term sequence of numbers.[5]

[4] Way back, we could have started by taking our fundamental language of arithmetic to be not L_A but L_A^+, i.e. the language you get by adding the exponential function to L_A. And, correspondingly, we could have taken as our basic theories Q$^+$ and PA$^+$, which you get from Q and PA by adding the obvious recursion axioms for the exponential. Then we'd have a very easily constructed β-function available and could have avoided the fuss in the rest of this section, and the need for the argument of the next footnote.

[5] Here is how to check that claim (this is just an exercise in elementary arithmetic, which is why we relegate it to a footnote for enthusiasts). First some notation and jargon. We write $a = rm(c, d)$ when a is the remainder when c is divided by d. We write D for a sequence of n numbers $d_0, d_1, d_2, \ldots d_n$ which are *relatively prime*, i.e. no two of them have a common factor other than 1. We write $Rm(c, D)$ for the sequence of remainders $rm(c, d_0)$, $rm(c, d_1)$, $rm(c, d_2)$, $\ldots$, $rm(c, d_n)$. And we put $|D|$ for the product $d_0 \cdot d_1 \cdot d_2 \cdot \ldots \cdot d_n$. Then we have

> *The Chinese Remainder Theorem* For any sequence D, then as c runs from 0 to $|D| - 1$, the sequences $Rm(c, D)$ are all different from each other.

But now note that the concept of a remainder on division can be elementarily defined in terms of multiplication and addition. Thus consider the following:

$$\mathsf{B(c, d, i, y)} =_{\text{def}} (\exists \mathsf{u} \leq \mathsf{c})[\mathsf{c} = \{\mathsf{S(d \times Si) \times u}\} + \mathsf{y} \ \wedge \ \mathsf{y} \leq (\mathsf{d \times Si})]$$

This, as we want, expresses our Gödelian β-function in L_A (and shows that it is a Δ_0 function).

11.4 L_A can express all p.r. functions: finishing the proof

Continuing the proof sketch for (3) Suppose we have some three-place β-function to hand. So, given any sequence of numbers $k_0, k_1, \ldots, k_x$, there are code numbers c, d such that for $i \leq x$, $\beta(c, d, i) = k_i$. Then we can reformulate

A. There is a sequence of numbers $k_0, k_1, \ldots, k_x$ such that: $k_0 = 1$, and if $u < x$ then $k_{Su} = Su \times k_u$, and $k_x = y$,

as follows:

B. There is some pair c, d such that: $\beta(c, d, 0) = 1$, and if $u < x$ then $\beta(c, d, Su) = Su \times \beta(c, d, u)$, and $\beta(c, d, x) = y$.

But we've seen that there's a particular Gödelian β-function which can be expressed in L_A by the open wff we abbreviated B. So fixing on this β-function, we can translate (B) into L_A as follows:

Proof Suppose otherwise. Then there are numbers $0 \leq c_1 < c_2 < |D|$, such that $Rm(c_1, D) = Rm(c_2, D)$. Put $c = c_2 - c_1$. Trivially, $c < |D|$. Now, it's another trivial fact that if c_1 and c_2 leave the same remainder when divided by some d, then c must exactly divide by d. So, since – by hypothesis – c_1 and c_2 leave the same remainders for each d_i in the sequence D, c divides by each d_i. And since the d_i have no factors in common that means that c must divide by their product $|D|$, contradicting the fact that $c < |D|$. ☒

Now, there are d_0 different possible remainders a number might have when divided by d_0 (i.e. $0, 1, 2, \ldots d_0 - 1$), d_1 possible remainders when divided by d_1, and so on. So there are $|D|$ different possible sequences of remainders $Rm(c, D)$. Hence, by our theorem, as c runs from 0 to $|D| - 1$, we get *every* possible sequence of remainders.

And now we can use this to show Gödel's claim that for any $k_0, k_1, \ldots, k_n$, we can find a pair of numbers c, d such that for $i \leq n$, $\beta(c, d, i) = k_i$, where $\beta(c, d, i) = rm(c, d(i + 1) + 1)$.

Proof Put s to be greatest of $n, k_0, k_1, \ldots, k_n$. Put $d = s!$ Then first note that for $0 \leq i \leq n$ the numbers $d_i = d(i + 1) + 1$ are relatively prime. For suppose otherwise, i.e. for some j, k where $1 \leq j < k \leq n + 1$, $dj + 1$ and $dk + 1$ have a common prime factor p. Plainly, $p > s$ (since any number up to s leaves a remainder 1 when dividing $s!j + 1$). But also since p divides $dj + 1$ and $dk + 1$, it divides their difference $d(k - j)$. But p can't divide d because it then wouldn't divide $dj + 1$. So p must divide $(k - j)$, which is less than n and so less than s. So $p < s$ Contradiction!

Thus the d_i are relatively prime. So by the Chinese Remainder Theorem, as we run through the sequences of remainders $Rm(c, D)$ for $c = 0$ to $|D| - 1$ we get every possible different sequence of remainders. And one of these sequences must be $k_0, k_1, \ldots, k_n$ (because each of those k_i is less than s so is a potential remainder on division by the corresponding d_i). ☒

C. $\exists \mathsf{c} \exists \mathsf{d}\{\mathsf{B}(\mathsf{c},\mathsf{d},0,1) \wedge$
$(\forall \mathsf{u} \leq \mathsf{x})[\mathsf{u} \neq \mathsf{x} \rightarrow \exists \mathsf{v} \exists \mathsf{w}\{(\mathsf{B}(\mathsf{c},\mathsf{d},\mathsf{u},\mathsf{v}) \wedge \mathsf{B}(\mathsf{c},\mathsf{d},\mathsf{Su},\mathsf{w})) \wedge \mathsf{w} = \mathsf{Su} \times \mathsf{v}\}] \wedge$
$\mathsf{B}(\mathsf{c},\mathsf{d},\mathsf{x},\mathsf{y})\}$

Abbreviate all that by '$\mathsf{F}(\mathsf{x},\mathsf{y})$', and we've arrived! For this evidently expresses the factorial function.

Let's summarize so far. We first noted that the p.r. definition of the factorial $n!$ tells us that there is a sequence of $(n+1)$ numbers satisfying a certain condition. Then we used the elegant β-function trick to re-write this as the claim that there is a code number for the sequence – or rather, two code numbers – satisfying a related condition. Using Gödel's particular β-function, we can then render this re-written version into L_A to give us a wff which expresses the recursive definition of the factorial.

So to finish the proof for (3), we just need to show that we can use the same β-function trick to express *any* function f defined by recursion from functions g and h which are already expressible in L_A.

Concluding the proof sketch for (3) Here, just for the record, is the entirely routine generalization we need. Suppose the function f is defined from the functions g and h by the standard p.r. recursion equations:

$$f(\vec{x},0) = g(\vec{x})$$
$$f(\vec{x},Sy) = h(\vec{x},y,f(\vec{x},y))$$

This definition amounts to fixing the value of $f(\vec{x},y) = z$ thus:

A* There is a sequence of numbers $k_0, k_1, \ldots, k_y$ such that: $k_0 = g(\vec{x})$, and if $u < y$ then $k_{u+1} = h(\vec{x},u,k_u)$, and $k_y = z$.

So using a three-place β-function again, that comes to

B* There is some c, d, such that: $\beta(c,d,0) = g(\vec{x})$, and if $u < y$ then $\beta(c,d,Su) = h(\vec{x},u,\beta(c,d,u))$, and $\beta(c,d,y) = z$.

Suppose we can already express the n-place function g by a $(n+1)$-variable expression G, and the $(n{+}2)$-variable function h by the $(n{+}3)$-variable expression H. Then – using '$\vec{\mathsf{x}}$' to indicate a suitable sequence of n variables – (B*) can be rendered into Q by

C* $\exists \mathsf{c} \exists \mathsf{d}\{\exists \mathsf{k}[\mathsf{B}(\mathsf{c},\mathsf{d},0,\mathsf{k}) \ \wedge \ \mathsf{G}(\vec{\mathsf{x}},\mathsf{w})] \ \wedge$
$(\forall \mathsf{u} \leq \mathsf{y})[\mathsf{u} \neq \mathsf{y} \rightarrow \exists \mathsf{v} \exists \mathsf{w}\{(\mathsf{B}(\mathsf{c},\mathsf{d},\mathsf{u},\mathsf{v}) \wedge \mathsf{B}(\mathsf{c},\mathsf{d},\mathsf{Su},\mathsf{w})) \wedge \mathsf{H}(\vec{\mathsf{x}},\mathsf{u},\mathsf{v},\mathsf{w})\}] \wedge$
$\mathsf{B}(\mathsf{c},\mathsf{d},\mathsf{y},\mathsf{z})\}$

Abbreviate this defined wff $\varphi(\vec{\mathsf{x}},\mathsf{y},\mathsf{z})$; it is then evident that φ will serve to express the p.r. defined function f. Which gives us the desired result (3). ⊠

So, we've shown how to established (1), (2) and (3). But this amounts, as we wanted, to a proof that every p.r. function can be expressed in L_A. We're done: Theorem 14 is in the bag!

11.5 The p.r. functions are Σ_1

Reviewing the proof we've just given, it's fairly easy to see that we've in fact already got all the materials to hand to show something stronger – namely that every p.r. function can be expressed in *by a strictly* Σ_1 *wff.* Hence

Theorem 15 *Every p.r. function is* Σ_1.

Before giving the official argument, here's the basic idea. We've just seen how to build up wffs for expressing p.r. functions with more and more complex definitions. We start with the Δ_0 initial functions. Compositions are expressed using existential quantifiers: so they don't take us beyond what can be expressed with Σ_1 wffs. And functions defined by recursion are expressed by wffs like (C*) . Now, (C*) does have a lot of existential quantifiers inside it (including some buried inside the embedded wffs G and H). But we can – using a simple little trick – drag *all* those internal existential quantifiers to the front, ending up with a Σ_1 wff which still expresses the same function as (C*). So defining functions by recursion applied to other Σ_1 functions still keeps us inside the class of Σ_1 functions.

Now we'll check those claims (but by all means skip the rest of this section with the fiddly details: we are just joining up the dots). To prove our theorem it is enough to show the following:

1′. The initial functions are Σ_1.

2′. If the functions g and h are Σ_1, then so is the function f defined by composition from g and h.

3′. If the functions g and h are Σ_1, then so is the function f defined by primitive recursion from g and h.

Proof for (1) We saw that the initial functions can be expressed using Δ_0 wffs, hence are Δ_0 functions. But as we noted at the end of Section 7.3, every Δ_0 function is trivially a Σ_1 function too. ☒

Proof for (2) Let's suppose, to take a simple case, that g and h are Σ_1 one-place functions, expressed by the strictly Σ_1 wffs $\mathsf{G(x,y)} = \exists\mathsf{uG(x,y,u)}$ and $\mathsf{H(x,y)} = \exists\mathsf{vH(x,y,v)}$ respectively, where G and H are Δ_0.

Now suppose f is defined by composition, so $f(m) = g(h(m))$. Then, f is expressed by $\exists\mathsf{z(G(x,z)} \wedge \mathsf{H(z,y))}$, i.e. $\exists\mathsf{z(}\exists\mathsf{uG(x,z,u)} \wedge \exists\mathsf{vH(z,y,v))}$. But that's equivalent to $\exists\mathsf{z}\exists\mathsf{u}\exists\mathsf{v(G(x,z,u)} \wedge \mathsf{H(z,y,v))}$ which is another strictly Σ_1 wff. So f can be expressed by a strictly Σ_1 wff, which was to be shown.

The argument now generalizes in the obvious ways to (i) more complex cases of composition, and (ii) cases where the Σ_1 functions being compounded are expressed by wffs with more than one existential quantifier at the front. ☒

Proof sketch for (3) Here's a preliminary observation to introduce the simple 'quantifier shift' trick we now need. Take the sample wff

i. $(\forall \mathsf{u} \leq \mathsf{n})\exists \mathsf{v}\mathsf{K}(\mathsf{u}, \mathsf{v})$.

If (i) is true, then for each $u \leq n$, there is a corresponding witness w_u which makes $\mathsf{K}(\mathsf{u}, \mathsf{w}_u)$ true. Now, take w to be the largest of those $n + 1$ witnesses w_u. Then $(\forall \mathsf{u} \leq \mathsf{n})(\exists \mathsf{x} \leq \mathsf{w})\mathsf{K}(\mathsf{u}, \mathsf{x})$ is true. And therefore

ii. $\exists \mathsf{v}(\forall \mathsf{u} \leq \mathsf{n})(\exists \mathsf{x} \leq \mathsf{v})\mathsf{K}(\mathsf{u}, \mathsf{x})$

is true. So if (i) is true, so is (ii); and obviously if (ii) is true, so is (i). Hence we can find a wff which expresses the same as (i) – because it is true just when (i) is – but which brings the *unbounded* existential quantifier $\exists \mathsf{v}$ out in front of the bounded universal quantifier $(\forall \mathsf{u} \leq \mathsf{n})$, leaving behind – as it were, as its shadow – a new *bounded* existential quantifier.[6]

Now to make use of this quantifier shift trick. Suppose f is defined by recursion from the Σ_1 functions g and h which are expressed by $\mathsf{G}(\vec{\mathsf{x}}, \mathsf{w})$ and $\mathsf{H}(\vec{\mathsf{x}}, \mathsf{u}, \mathsf{v}, \mathsf{w})$, where both those wffs are strictly Σ_1. Then, as we saw, f is expressed by the corresponding

$$\begin{aligned} \text{C*}\quad & \exists \mathsf{c}\exists \mathsf{d}\{\exists \mathsf{k}[\mathsf{B}(\mathsf{c}, \mathsf{d}, 0, \mathsf{k}) \land \mathsf{G}(\vec{\mathsf{x}}, \mathsf{w})] \land \\ & \quad (\forall \mathsf{u} \leq \mathsf{y})[\mathsf{u} \neq \mathsf{y} \to \exists \mathsf{v}\exists \mathsf{w}\{(\mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{u}, \mathsf{v}) \land \mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{Su}, \mathsf{w})) \land \mathsf{H}(\vec{\mathsf{x}}, \mathsf{u}, \mathsf{v}, \mathsf{w})\}] \land \\ & \qquad \mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{y}, \mathsf{z})\} \end{aligned}$$

where B, remember is Δ_0. So now consider the wff (C**) constructed as follows. First we drag the quantifier $\exists \mathsf{k}$ and any existential quantifiers embedded in G to the front.[7] We then use the quantifier shift trick to drag the quantifers $\exists \mathsf{u}\exists \mathsf{v}$ plus any existential quantifiers embedded in H to the front, moving them past the bounded universal $(\forall \mathsf{u} \leq \mathsf{y})$, leaving behind bounded existential quantifiers as their shadows. The resulting open wff (C**) will then have a block of existential quantifiers at the front, followed by a Δ_0 kernel. So it follows that (C**) is strictly Σ_1, while it still expresses just the same function f as (C*) as it is satisfied by just the same numbers.

Hence, as we wanted, we've shown that a function defined by recursion from Σ_1 functions is itself Σ_1. ⊠

11.6 The adequacy theorem

Now we can simply put together the Big Results Theorem 13 (established in Section 11.1) and Theorem 15.

Theorem 13 Q *can capture any* Σ_1 *function as a function,*

Theorem 15 *Every p.r. function is* Σ_1,

[6] NB: To drag an existential quantifier forwards across an *unbounded* universal quantifer is to commit a horrible quantifier shift fallacy. But here we are dragging across a *bounded* universal quantifier, and that makes all the difference!

[7] Changing variables, of course, if that's what it takes to avoid clashes.

to get the Really Beautiful Big Theorem that we've been aiming for.[8] Q can capture all p.r. functions as functions, i.e.

Theorem 16 Q *is p.r. adequate.*

And this implies that Q can capture every p.r. property and relation (and by a Σ_1 wff). That's because a property is p.r. if it has a p.r. characteristic function; and this characteristic function, being p.r., can be captured in Q. But by the trivial result we noted in Section 10.4, if a property's characteristic function is capturable by the Σ_1 wff $\varphi(\mathsf{x}, \mathsf{y})$, so is that property itself by the Σ_1 wff $\varphi(\mathsf{x}, \mathsf{0})$. Likewise for relations.

Since PA can prove everything Q proves, that (of course) means that PA is p.r. adequate too. And it is worth noting that we have in fact proved somewhat more than we set out to do. For we've shown not only that every p.r. function is captured in Q and PA, but that every such function is captured by a wff which reveals it to *be* a p.r. function.

To repeat once more, a p.r. function f can always be defined by a chain of definitions by composition and primitive recursion, starting from some initial functions. And we've just shown that we can express and indeed capture f by a corresponding wff *which is built up by steps which recapitulate the definitions.* Then we can refine our theorem: Q and PA can capture every p.r. function in this kind of perspicuous way. (And we'll take it henceforth that when we talk of a p.r. function being captured by a wff in a formal arithmetic, we have in mind this sort of revealing representation.)

[8] Of course, Gödel in 1931 didn't know about Q, which was first isolated as a minimal p.r. adequate arithmetic in 1952. So Gödel didn't himself have the theorem in our form. He *did*, however, indicate a proof of the adequacy of a richer system, using the absolutely key β-function trick: and it was then relatively (though only relatively!) easy to show that the proof goes through even in such a weak system as Q.

Interlude: a very little about *Principia*

In the last Interlude, we gave a five-stage map of our route to Gödel's First Incompleteness Theorem. The first two stages we mentioned (namely, looking at Q and PA, then defining the p.r. functions and proving Q's p.r. adequacy) are now behind us. We have already mentioned one neat idea from Gödel's epoch-making 1931 paper, the β-function trick; but most of his proof is still ahead of us – and at the end of this Interlude, we'll review the stages that remain. But before getting down to details, let's pause to take in a little scene-setting historical background.

(a) We'll say rather more about the historical context in a later Interlude on Hilbert's Programme. For now, let's just say enough to explain at least the *title* of Gödel's great paper, 'On formally undecidable propositions of *Principia Mathematica* and related systems I'.[1] What is being referred to here?

As we noted in Section 8.5, Frege aimed in *The Basic Laws of Arithmetic* to reconstruct arithmetic on a secure footing by deducing from logic plus definitions. But – in its original form – his overall logicist project flounders on Frege's fifth Basic Law, which postulates the existence of so many classes as to lead to contradiction. And the fatal flaw that Russell exposed in Frege's system was not the only contradiction to beset early treatments of the theory of classes (Georg Cantor had already found other paradoxes).

Various responses to these paradoxes were proposed at the beginning of the twentieth century. One suggestion is, in effect, to keep much of Frege's logic but to avoid making the further move that gets him into disastrous trouble.

To explain: Frege's general logical system involves a kind of *type hierarchy.* It very carefully distinguishes 'objects' (things, in a broad sense) from properties from properties-of-properties from properties-of-properties-of-properties, etc, and insists that every item belongs to a determinate level of the hierarchy. Then the claim is – plausibly enough – that it only makes sense to attribute properties which belong at level l to items at level $l - 1$. For example, the property of *being wise* is a level 1 property, while Socrates is an item at level 0; and it makes sense to attribute that property to Socrates, i.e. to claim that Socrates is wise. Likewise, the property of *having some instances* is a level 2 property, and it makes sense to attribute that property to the level 1 property of being wise, i.e. to claim that the property of being wise has some instances. But you get nonsense if, for example, you try to attribute that level 2 property to Socrates

[1] That's a roman numeral one at the end of the title! Gödel originally planned a Part II, fearing that readers would not, in particular, accept the very briskly sketched Second Theorem without further elaboration. But Gödel's worries proved groundless and Part II never appeared.

and claim that Socrates has some instances.

Note that this strict stratification of items into types blocks the derivation of the property analogue of Russell's paradox about classes. The original paradox, recall, concerned the class of all classes that are not members of themselves. So now consider the putative property of *being a property that doesn't apply to itself.* Does this apply to itself? It might seem that the answer is that it does if it doesn't, and it doesn't if it does – contradiction! But on Frege's hierarchical theory of properties, there is no real contradiction to be found here: (i) Every genuine level l property belongs to some particular level of the hierarchy, and only applies to items at the next level down. A level l property therefore can't sensibly be attributed to any level l property, including itself. (ii) However, there is no generic property of 'being a property that doesn't apply to itself' shared by every property at any level. No genuine property can be type-promiscuous in that way.

One way to avoid class-theoretic paradox, then, is to stratify the universe of classes into a type-hierarchy in the way that Frege stratifies the universe of properties. So suppose we now distinguish classes from classes-of-classes from classes-of-classes-of-classes, and so forth; and on one version of this approach we then insist that classes at level l can only have as members items at level $l - 1$.[2] Frege himself doesn't take this line: his disastrous Basic Law V in effect flattens the hierarchy for classes and puts them all on the same level. However, Bertrand Russell and Alfred North Whitehead do in a sense adopt the hierarchical view of classes in their monumental *Principia Mathematica* (1910–13). They retain and develop Frege's stratification of properties and then link this to the stratification of classes in a very direct way, by treating talk about classes as in effect just lightly disguised talk about their corresponding defining properties. The resulting system is – as far as we know – consistent.

(b) Having established their paradox-blocking logical framework, Russell and Whitehead set out in *Principia* – like Frege in his *Basic Laws*, and following a broadly similar strategy – to derive all of arithmetic from definitional axioms.[3] Indeed, the project is even more ambitious: the ultimate aim (as Russell described it a decade earlier) is to prove that

> *all* mathematics deals exclusively with concepts definable in terms of a very small number of logical concepts, and ... *all* its propositions are deducible from a very small number of fundamental logical principles. (Russell, 1903, p. xv, my emphases.)

But let's concentrate on the more modest but still ambitious project of deriving just arithmetic from logic plus definitions.

[2] An alternative approach – the now dominant Zermelo-Fraenkel set theory – is more liberal: it allows sets formed at level l to contain members from *any* lower level. In the jargon, we get a *cumulative* hierarchy. But this is still enough to block paradox.

[3] Compare the intuitively appealing project we described right at the outset, in Section 1.1.

This isn't the place to review the details and differences of the Frege-Russell constructions, their successes and its failures. Still, for those who haven't encountered this logicist project before, perhaps we should give a quick taster of a few of the ingredients involved, so you get *some* sense of how the dish might be cooked. So . . .

i. We'll say that the Fs and Gs are *equinumerous* just in case there is a one-one correspondence between the Fs and the Gs. To take a hackneyed example, the knives and forks are equinumerous if you can pair them up, one to one, with none left over.

   Now, the idea of there being a one-one correspondence between the Fs and the Gs surely *is* a logical one: it can be defined using quantifiers and identity. In words: there's such a correspondence if there is a relation R such that every F has relation R to a unique G, and for every G there is a unique F which has relation R to it. In symbols:

   $$\exists R\{\forall x(Fx \rightarrow \exists! y(Gy \wedge Rxy)) \wedge \forall y(Gy \rightarrow \exists! x(Fx \wedge Rxy))\}$$

   Here, '∃!' is the familiar uniqueness quantifier (see Section 10.2, fn. 2); and the initial quantifier is a second-order quantifier ranging over two-place relations.

ii. Intuitively, the number of Fs is identical to the number of Gs just in case the Fs and Gs are equinumerous in the logical sense just defined. This claim is nowadays – with only tenuous justification – called *Hume's Principle*. Any attempt to identify the numbers should surely respect it.

iii. Here's another, equally intuitive, claim – call it the *Successor Principle*: the number of Fs is the successor of the number of Gs just in case there is an object o which is an F, and the remaining things which are F-but-not-identical-to-o are equinumerous with the Gs.

iv. What though *are* numbers? Here's a brute-force way of identifying them while respecting Hume's Principle. Take *the number of Fs* to be *the class of all classes equinumerous with the class of Fs*, where classes are equinumerous, of course, if their members are equinumerous. Then, as we want, the number of Fs is identical with the number of Gs just if the class of all classes with as many members as there are Fs is identical with the class of all classes with as many members as there are Gs, which holds just if the Fs and Gs are indeed equinumerous.

v. Taking this brute-force line on identifying numbers, we can immediately define *zero* to be the class of all classes equinumerous with the non-self-identical things. For assuredly, zero *is* the number of x such that $x \neq x$. And, on the most modest of assumptions, zero will then exist – it is the class of all empty classes; but there is only one empty class since classes with the same members are the same class; so zero is the class of *the* empty class.

vi. And now – a very cunning trick! – let's define *one* to be the class of all classes equinumerous with the class containing just zero. Which makes it the case that one *is* the number of x such that $x = 0$. And also makes it the case that one is the successor of zero. Likewise, we can now go on to define *two* to be the class of all classes equinumerous with the class containing just zero and one. Which makes it the case that two *is* the number of x such that $x = 0 \lor x = 1$. We can go on to define *three* to be the class of all classes equinumerous with the class containing just zero and one and two. Which makes it the case that three *is* the number of x such that $x = 0 \lor x = 1 \lor x = 2$. And so it goes.

vii. Finally, we need an account of what the finite *natural* numbers are (for note that our basic definition of *the number of Fs* applies equally when there is an infinite number of Fs). Well, let's say that a property F is *hereditary* if, whenever a number has it, so does its successor. Then *a number is a natural number if it has all the hereditary properties that zero has.* Which in effect defines the natural numbers as those for which the familiar induction principle holds.

We have a story, then, about what numbers themselves are. We have a story about zero, one, two, three and so on. We have a story about what it is for one number to be the successor of another (you can readily check e.g. that one is the successor of zero and two the successor of one etc. by our definition of succession). We have a story about which numbers are natural numbers (again, you can check that one, two, three and so on are natural numbers on our definition). So suppose that you buy the (big!) assumption that the talk about 'classes' in the story so far still counts as *logical* talk, broadly construed. Then we are at least launched on our way towards (re)constructing arithmetic in logical terms. And the logicist hopes to continue the story in a way that would reveal *all* arithmetical truths to be derivable (in a consistent system!) from what could be regarded as broadly logical apparatus plus definitions.

Now, you might well wonder, for example, whether the cunning trick that gets us the natural number sequence is a bit too cunning: you might think it smacks of conjuring the numbers into existence. However, it would take us far to far afield to pause to consider whether the logicist project already founders at this point. I just hope to have said enough to give you a hint of how Frege and the authors of *Principia* could sensibly think that there was a possible enterprise here. But now enter Gödel ....

(c) What the First Incompleteness Theorem shows is that, despite its great power, Russell and Whitehead's construction still can't capture even all truths of basic arithmetic, at least assuming it *is* consistent. As Gödel puts it in the opening words of his paper:

> The development of mathematics toward greater precision has led, as is well known, to the formalization of large tracts of it, so that one can prove any theorem using nothing but a few mechanical rules. The most comprehensive formal systems that have been set up hitherto yet are the system of *Principia Mathematica* on the one hand and the Zermelo-Fraenkel axiom system for set theory ... on the other. These two systems are so comprehensive that in them all methods of proof today used in mathematics are formalized, that is, reduced to a few axioms and rules of inference. One might therefore conjecture that these axioms and rules of inference are sufficient to decide *any* mathematical question that can at all be formally expressed in these systems. It will be shown below that this is not the case, that on the contrary there are in the two systems mentioned relatively simple problems in the theory of integers which cannot be decided on the basis of the axioms. This situation is not in any way due to the special nature of the systems that have been set up, but holds for a very wide class of formal systems; .... (Gödel, 1931, p. 145)

Now, to repeat, Russell and Whitehead's system is built on a logic that allows quantification over properties, properties-of-properties, properties-of-properties-of-properties, and so on up the hierarchy. Hence the language of *Principia* is *immensely* richer than the language L_A of first-order PA (where we can only quantify over individuals, and which has no way of representing properties-of-properties-of-properties or higher types). It perhaps wouldn't be a great surprise, then, to learn that Russell and Whitehead's modest collection of axioms doesn't settle every question that can be posed in their very rich formal language. What *is* a great surprise is that there are 'relatively simple' propositions which are 'formally undecidable' in *Principia* – by which Gödel means just that there are wffs φ of basic arithmetic such that we can't prove either φ or $\neg\varphi$ from the axioms. Even if we buy all the assumptions of *Principia*, and can e.g. quiet our worries about the appearance of a conjuring trick in constructing the number series, we still don't get what the logicist hoped to get, i.e. a complete theory of arithmetic. And similarly, there are arithmetical propositions which are 'formally undecidable' in ZF set theory.

(d) As Gödel himself notes, his incompleteness proof only needs to invoke some fairly elementary features of the full-blooded theories of *Principia* and of ZF, and these features are equally shared by PA. So let's now forget about *Principia*: it will do little harm, for our purposes, to indulge henceforth in a historical fiction and pretend that Gödel was really talking about PA all along.

In what follows, there are also some other deviations from the details of his original proof; but the basic lines of argument in the next three chapters are all in his great paper. Not surprisingly, other ways of establishing his results (and generalizations and extensions of them) have been discovered since 1931, and we will be mentioning some of these later. But there remains a good deal to be said for introducing the incompleteness theorems by something close to Gödel's own arguments.[4]

(e) Here, then, is an abbreviated reminder of the three stages in our Gödelian proof which remain ahead of us:

1. Next, we look at Gödel's great innovation – the idea of systematically associating expressions of a formal arithmetic with numerical codes. We'll stick closely to Gödel's original type of numbering scheme. With a coding scheme in place, we can reflect key properties and relations of strings of symbols of PA (to concentrate on that theory) by properties and relations of their Gödel numbers. For a pivotal example, we can define the numerical relation *Prfseq*(m, n) which holds when m codes for a sequence of wffs that is a PA proof, and n codes the closed wff that is thereby proved. And Gödel proves that such arithmetical properties and relations are primitive recursive. (Chapter 12)

2. Since *Prfseq*(m, n) is p.r., it can be expressed – indeed, can be captured – in PA. We can now use this fact to construct a sentence G that, given the coding scheme, 'says' there is no number which is the Gödel number of a PA proof of the wff which results from a certain construction – where the wff which results is none other than G itself. So in effect G 'says' of itself 'I am unprovable in PA'. We can then show that G is indeed unprovable, assuming no more than that PA is consistent. So we've found an arithmetical sentence which is true but unprovable in PA. (And given a slightly stronger assumption than PA's consistency, $\neg$G must also be unprovable in PA.) Moreover, it turns out that this unprovable sentence is in one respect a pretty simple one: it is in fact a Π_1 wff. (Chapter 13)

3. As Gödel notes, the true-but-unprovable sentence G for PA is in fact generated by a method that can be applied to any other arithmetic that satisfies some modest conditions. Which means that PA is not only incomplete but incompletable. Indeed, *any* properly axiomatized that contains the weaker theory Q is incompletable. (Chapter 14)

So, to work . . . !

[4] Here's one small advantage of approaching things this way: it emphasizes that Gödel's 1931 incompleteness results do *not* depend on the general theory of what makes of a computable function (a general theory which didn't become settled until the later 1930s).

12 The arithmetization of syntax

We now introduce Gödel's simple but wonderfully powerful idea of associating *numbers* (the ostensible subject matter of the wffs of a formal arithmetic) with the *wffs* and *proofs* from a formal theory.

We'll fix on a particular coding scheme. This device enables us to correlate PA's expressions and proofs with code numbers. Then, corresponding to the syntactic property of being a wff, we can define the numerical property *Wff*, where *Wff*(n) holds when n is code number in our scheme for a wff of PA. And it will be easy to see that *Wff* will be primitive recursive.

More excitingly, we can define the numerical relation *Prfseq*(m, n) which holds just when m is the number in our scheme of a PA-proof of the sentence with number n. Moreover, it will also be easy to see – at least in an informal way – that this relation too is primitive recursive. So given what was shown in the last chapter, *Prfseq* will itself be capturable in PA.

We can also introduce the idea of the *diagonalization* of a wff. Roughly speaking, this is the idea of taking a wff $\varphi(\mathsf{y})$, and substituting its own code number in place of the free variable. Now, think of a code number as a way of referring to a wff. Then the operation of 'diagonalization' allows us to form a wff that (as it were) refers to itself. We will use this trick in the next chapter to form a Gödel sentence that encodes, roughly speaking, 'I am unprovable in PA'.

These, then, are the basically straightforward ideas to carry away from this chapter. However, we really ought to outline a proper proof of the key claim that *Prfseq* is primitive recursive. That's the business for the last section.

12.1 Gödel numbering

We've already encountered one numbering device in Section 3.5; we mapped symbols from the alphabet of a theory's language to (base ten) digits, and associated a concatenation of symbols with (the number expressed by) the corresponding concatenation of digits. This sort of thing would work for our present purposes too, but we'll in fact use something more like Gödel's original numbering scheme. We'll start by thinking about how to encode expressions of L_A.

Suppose that our version of L_A has the usual logical symbolism (connectives, quantifier symbols, identity, brackets), and symbols for zero and the successor, addition and multiplication functions: associate all those with odd numbers (different symbol, different number, of course). L_A also has an inexhaustible supply of variables, which we'll associate with even numbers. So, to pin that down, let's fix on this preliminary series of *symbol codes*:

$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$	$\forall$	$\exists$	$=$	(	)	0	S	$+$	$\times$	x	y	z	...
1	3	5	7	9	11	13	15	17	19	21	23	25	27	2	4	6	...

Our Gödelian numbering scheme for expressions e is now defined in terms of this table of preliminary symbol codes as follows:

> Let e be a sequence of $k+1$ symbols $s_0, s_1, s_2, \ldots, s_k$. Then its *Gödel number* (g.n.) is calculated by taking the symbols' correlated code-numbers, using them in turn as exponents for the first $k+1$ prime numbers $\pi_0, \pi_1, \pi_2, \ldots, \pi_k$, and then multiplying the results.

For example:

i. The single symbol 'S' has the g.n. 2^{23} (the first prime raised to the appropriate power as read off from our correlation table of symbol codes).

ii. The standard numeral SS0 has the g.n. $2^{23} \cdot 3^{23} \cdot 5^{21}$ (the product of the first three primes raised to the appropriate powers).

iii. The wff

$$\exists \mathsf{y}\,(\mathsf{S0} + \mathsf{y}) = \mathsf{SS0}$$

has the g.n.

$$2^{13} \cdot 3^{4} \cdot 5^{17} \cdot 7^{23} \cdot 11^{21} \cdot 13^{25} \cdot 17^{4} \cdot 19^{19} \cdot 23^{15} \cdot 29^{23} \cdot 31^{23} \cdot 37^{21}$$

That last number is, of course, *enormous*. So when we say that it is elementary to decode the resulting g.n. by taking the exponents of prime factors, we don't mean that the computation is quick and easy. We mean that the computational routine required for the task – namely, repeatedly extracting prime factors – involves no more than the mechanical operations of school-room arithmetic.

Three remarks. First, we've earlier allowed the introduction of abbreviatory symbols into PA's language (for example, '$\leq$' and '3'); take the g.n. of an expression including such symbols to be the g.n. of the unabbreviated version.

Second, we will later be assuming there are similar numbering schemes for the expressions of other theories with possibly different languages L. We can imagine each of these numbering schemes to be built up in the same way but from a different table of preliminary symbol codes to cope with the different basic symbols of L. We won't spell out the details.

Third, we are going to be introducing numerical properties like *Wff* and proving them them to be primitive recursive. But note, *Wff*(n) is to hold when n is the code number of an L_A wff *according to our Gödel numbering scheme*. However, our numbering scheme was fairly arbitrarily chosen: we could, for example, shuffle around the preliminary assignment of basic symbol codes to get a different numbering scheme, or (more radically) we could use a scheme that isn't based on powers of primes. So could it be that a property like *Wff* be p.r. when defined in terms of one arbitrarily chosen numbering scheme and not p.r. when defined in terms of some equally sensible scheme?

No. Arguing informally, suppose S_1 and S_2 are sensible numbering schemes. And suppose S_1 assigns code n_1 to a certain L_A expression, and S_2 assigns code n_2. Then consider the process of decoding n_1 to find the original expression and re-encoding to get n_2. If the coding schemes are anything at all like our simple-minded numbering device in Section 3.5 or the Gödelian scheme we've just introduced, then this process will involve a simple computation without any open-ended loopings. The computation can be done just using 'for' loops. Hence, there will be a *primitive recursive* function which maps n_1 to n_2, and similarly there will be another p.r. function which maps n_2 back to n_2. So, in the light of that informal argument, we can make the following formal stipulation. We'll say that a system of code numbering S is *acceptable* if there is a p.r. function tr which 'translates' code numbers under our official Gödelian scheme into code numbers under scheme S, and another reverse function tr^{-1} which converts code numbers under scheme S back into code numbers under our scheme.

With that stipulation, it is immediate that a property like *Wff* defined using our scheme is p.r. if and only if the corresponding property Wff_S defined using scheme S is p.r., for any acceptable scheme S. Why? Well, let the characteristic functions of *Wff* and Wff_S be *wff* and wff_S respectively. Then $wff_S(n) = tr(wff(n))$, hence wff_S will be p.r. by composition so long as *wff* is p.r.; and similarly $wff(n) = tr^{-1}(wff_S(n))$, hence *wff* is p.r. if wff_S is.

In sum, given our stipulation of what counts as an acceptable coding scheme, whether a property like *Wff* is p.r. is *not* dependent on our particular choice of scheme.

12.2 Coding sequences

As we've already flagged up, the relation $Prfseq(m, n)$ will be crucial to what follows, where this relation holds just when m codes for an array of wffs that is a PA proof, and n codes for the closed wff (sentence) that is thereby proved. But how *do* we code for proof-arrays?

The details will depend on the kind of proof-system we've adopted for our version of PA. To keep things simple, let's assume that the proof-system is a rather old-fashioned linear one (not a tree system), so proof-arrays are simply *sequences* of wffs. Then a nice way of coding these is by what we'll call *super Gödel numbers*. So, given a sequence of PA wffs or other expressions

$$e_0, e_1, e_2, \ldots, e_n$$

we first code each e_i by a regular g.n. g_i, to yield a resulting sequence of regular Gödel numbers

$$g_0, g_1, g_2, \ldots, g_n$$

We then encode this sequence of regular Gödel numbers into a *super g.n* by repeating the trick of taking powers of primes to get a single super g.n.

$$2^{g_0} \cdot 3^{g_1} \cdot 5^{g_2} \cdot \ldots \cdot \pi_n^{g_n}$$

Hence, decoding a super g.n. involves two steps of taking prime factors: first find the exponents of the prime factors of the super g.n.; then treat those exponents as themselves regular g.n., and take their prime factors to arrive back at a sequence of PA expressions.[1]

We can now define more carefully the relation *Prfseq* which is going to concern us so much:

> *Prfseq*(m, n) holds just if m is the *super* g.n. of a sequence of wffs that is a PA proof of a closed wff with *regular* g.n. n.

12.3 *Prfseq* is p.r.

We now need to convince ourselves of

> **Theorem 17** *Prfseq*(m, n) *is primitive recursive.*

And we have a much easier time of it than Gödel did. Writing at the very beginning of the period when concepts of computation were being forged, he certainly couldn't expect his audience to take anything on trust about what was or wasn't '*rekursiv*' or – as we would now put it – primitive recursive. He therefore had to do all the hard work of explicitly showing how to define *Prfseq*(m, n) by a long chain of definitions by composition and recursion.

However, assuming only a very modest familiarity with the ideas of algorithms and computer programs, we can perhaps short-cut all that effort and be entirely persuaded by the following:

A very sketchy argument To determine whether *Prfseq*(m, n), proceed as follows. First doubly decode m: that's a mechanical exercise. Now ask: is the result a sequence of PA wffs? That's algorithmically decidable (since it is decidable each separate string of symbols is a wff). If it does decode into a sequence of wffs, ask: is this sequence a properly constructed proof? That's decidable too (check whether each wff in the sequence is either an axiom or is an immediate consequence of previous wffs by one of the rules of inference of PA's logical system). If the sequence is a proof, ask: is the last wff in the sequence, the wff proved, a closed wff – i.e. does it lack free variables? That's decidable. If it is a sentence, ask: does that final wff have the g.n. n? That's again decidable. So, putting all that together, there is a computational procedure for telling whether *Prfseq*(m, n) holds.

[1] On our way of doing things, certain numbers can be both the regular g.n. of one expression and the super g.n. of a sequence of other expressions. No matter. We always make it clear in context whether a number is to be treated as one sort of code or the other.

Moreover, *at each and every stage, the computation involved is a straightforward, bounded procedure that doesn't involve any open-ended search.* In other words, suppose that we set out to construct a program for determining whether $Prfseq(m, n)$. Then we will be able to do this using programming structures no more exotic than bounded 'for' loops (in particular, we don't need to use any of those open-ended 'do while'/'do until' structures that take us outside the bounds of the primitive recursive). Now, most of the computations we've described involve shuffling strings of symbols; but under the bonnet, so to speak, that's really just more operations on binary numbers. And if the whole computation can therefore be done ultimately with 'for' loops operating on numbers, the numerical relation which is decided by the whole procedure must be primitive recursive (see Section 9.4). ⊠

That is indeed sketchy, but the argument may well strike you as quite convincing enough. And if you are therefore happy to take it on trust that we can make all this rigorous, that's just fine: if you aren't, then Section 12.7 begins to outline a proper proof.

12.4 Some cute notation

Before proceeding, let's introduce a really pretty bit of notation. Assume we are dealing with a theory couched in some language L which has standard numerals. And assume that we have chosen some system of Gödel-numbering. Then

> If φ is an L-expression, then we'll use '$\ulcorner\varphi\urcorner$' *in our logicians' augmented English* to denote φ's Gödel number.

Borrowing a species of quotation mark is appropriate because the number $\ulcorner\varphi\urcorner$ can be thought of as referring to the expression φ in our coding scheme.

We are also going to use this very same notation as a placeholder for numerals inside our formal language, so that (in our second usage)

> *In abbreviated L-expressions*, '$\ulcorner\varphi\urcorner$' is shorthand for L's standard numeral for the g.n. of φ.

In other words, inside formal expressions '$\ulcorner\varphi\urcorner$' stands in for the numeral for the number $\ulcorner\varphi\urcorner$.

A simple example to illustrate:

1. '$\mathsf{SS0}$' is an L_A expression, the standard numeral for 2.

2. On our numbering scheme $\ulcorner\mathsf{SS0}\urcorner$, the g.n. of '$\mathsf{SS0}$', is $2^{21} \cdot 3^{21} \cdot 5^{19}$.

3. So, by our further convention, we can also use the expression '$\ulcorner\mathsf{SS0}\urcorner$' inside (a definitional extension of) L_A, as an abbreviation for the standard numeral for that g.n., i.e. as an abbreviation for '$\mathsf{SSS}\ldots\mathsf{S0}$' with $2^{21} \cdot 3^{21} \cdot 5^{19}$ occurrences of 'S'!

This double usage – outside a formal language to denote a g.n. of a formal expression and inside a formal language to take the place of a standard numeral for that g.n. – is a fairly common convention and should by this stage cause no confusion at all.

12.5 The idea of diagonalization

Gödel is going to tell us how to construct a formal wff G in PA that encodes 'I am unprovable'. We now have an inkling of how he can do that: wffs can contain numerals which refer to Gödel-numbers which are in turn correlated with wffs.

So Gödel's construction involves taking a particular open wff that we'll abbreviate U, or by $\mathsf{U(y)}$ when we want to emphasize that it contains just 'y' free. This wff has g.n. $\ulcorner\mathsf{U}\urcorner$. And then Gödel substitutes *the numeral for* U*'s g.n.* for the free variable in U. So the key step involves forming the wff $\mathsf{U}(\ulcorner\mathsf{U}\urcorner)$.

This involves something quite closely akin to the 'diagonal' constructions we encountered in e.g. Sections 5.2 and 9.5. In the first of those cases, we matched the index of a wff $\varphi_n(\mathsf{x})$ (in an enumeration of wffs with one free variable) with the numeral substituted for its free variable, to form $\varphi_n(\mathsf{n})$. In the second case, we matched the index of a function f_n (in an enumeration of p.r. functions) with the number the function takes as argument, to form $f_n(n)$. Here, in our Gödelian 'diagonal' construction, we match U's Gödel number – and we can think of this as indexing the wff in a list of wffs – with the numeral substituted for its free variable, and this yields the Gödel sentence G.

Now just note the following additional point. Given the wff U, it can't matter much whether we do the diagonalization construction by forming (i) $\mathsf{U}(\ulcorner\mathsf{U}\urcorner)$ (as Gödel himself does) or by forming (ii) $\exists\mathsf{y}(\mathsf{y} = \ulcorner\mathsf{U}\urcorner \wedge \mathsf{U(y)})$. For (i) and (ii) are trivially equivalent. But it makes a few things go very slightly easier if we do things the second way.

So with that motivation, here's an official definition:[2]

> The *diagonalization* of φ is $\exists\mathsf{y}(\mathsf{y} = \ulcorner\varphi\urcorner\ \wedge\ \varphi)$.

12.6 *Gdl* and *diag* and are p.r.

(a) Diagonalization is, evidently, a very simple mechanical operation on expressions. So there will be a corresponding simple computable function dealing with numerical codes for expressions which 'tracks' the operation. In other words,

> There is a p.r. function $diag(n)$ which, when applied to a number n which is the g.n. of some wff, yields the g.n. of that wff's diagonalization.

[2] By the way there is no special significance to using the variable 'y' for the free variable here! But we'll keep this choice fixed, simply for convenience. Note our official definition of diagonalization applies to any wff, whether it contains 'y' free or not.

Another very sketchy argument Decode the g.n. $n = \ulcorner\varphi\urcorner$ to get some expression φ. Then form its diagonalization, $\exists \mathsf{y}(\mathsf{y} = \ulcorner\varphi\urcorner \wedge \varphi)$. Then work out the g.n. of the result. This very simple mechanical procedure to compute $diag(n)$ doesn't involve any unbounded searches. So we again we will be able to program the procedure using just 'for' loops. Hence *diag* is p.r. ⊠

(b) Now consider the following relation which will play a starring role in the next chapter:

$$Gld(m, n) =_{\text{def}} Prfseq(m, diag(n))$$

When does this relation hold? Evidently,

> $Gld(m, n)$ holds when m is the super g.n. for a PA proof of the diagonalization of the wff with g.n. n.

Again, it is easy to see that *Gld* is intuitively decidable by a computation without unbounded searches: so we'd expect it to be primitive recursive. Given that *Prfseq* and *diag* are indeed p.r., *Gld* is p.r. by composition.[3]

12.7 Proving that *Prfseq* is p.r.

We have given a very informal but hopefully persuasive argument for Theorem 17, the claim that *Prfseq* is primitive recursive. But Gödel, as we said, gives a cast-iron proof of this by showing how to define a sequence of more and more complex functions and relations by composition and recursion. Inevitably, this is a laborious job: Gödel does it with masterly economy and compression but, even so, it takes him 45 steps of function-building to show that *Prfseq* is p.r.

We've in fact already traced some of the first steps in Section 9.7. We showed, in particular, that extracting exponents of prime factors – the key operation used in decoding Gödel numbers – involves the p.r. function *exp*. We now need to keep going in the same vein, defining ever more complex functions. What I propose to do here is fill in the next few steps moderately carefully, and then indicate much more briefly how the rest go. This should be just enough to give you a genuine feel for Gödel's demonstration and convince you that it *can* be completed, without going into too much horrible detail.[4]

However, although we are only going to give a partial proof that *Prfseq* is p.r., by all means skip even this cut-down discussion, and jump to the next chapter where the real excitement starts. You'll miss nothing of wider conceptual interest. Gödel's Big Idea is that *Prfseq* is p.r.: checking that this Big Idea is right just involves cracking some brain-teasers. (Look at it like this. We argued on general grounds in the Section 12.3 that *Prfseq* is p.r.: hence there must be a LOOP

[3] Strictly: the characteristic function of *Gld* is definable by composition from the characteristic function of *Prfseq* and the function *diag*. Check that claim!

[4] Masochists and 'completists' are quite welcome to struggle through e.g. (Mendelson, 1997, pp. 193–198).

program for determining whether $Prfseq(m, n)$ holds for particular numbers m, n. This section in effect begins to describe how to write the program. Which is fun in its way, if you like that kind of thing. But once you are are convinced that the programming tricks *can* be done, you can cheerfully forget *how* they are done.)

Here's a local road-map. First, in (a), we introduce a p.r. function that takes us from the codes of two expressions to the code for their concatenation. This plays a key role in what follows. Then (b) we put this function to work in a preliminary way. In (c) we prove that the relation *Termseq* is p.r., where $Termseq(m, n)$ holds just when m is the super g.n. of a sequence of expressions which show how the expression with g.n. n can be built up from '$\mathsf{0}$' and/or variables using successor, addition and multiplication. In other words, $Termseq(m, n)$ holds when m codes the constructional history of the term with code n.

Now (d) this idea of coding up the constructional history of a term can mirrored by the idea of coding up the constructional history of a wff, and then (e) mirrored again by the idea of coding up a sequence of wffs that forms a proof. And that's what we need to show that $Prfseq(m, n)$ is indeed primitive recursive.

(a) *The concatenation function* If you are still reading, first quickly revisit Section 9.7. Recall two facts in particular. Keeping the old numbering,

6. The function $exp(n, i)$ is p.r., where this returns the exponent of π_i in the prime factorization of n.

7. The function $len(n)$ is p.r., where this returns the number of distinct prime factors of n.

Note that if n is the g.n. of an expression e which is a sequence of symbols $s_0, s_1, s_2, \ldots, s_k$, then $exp(n, i)$ gives the symbol code of s_i. And if n is the super g.n. of a sequence of wffs or other expressions $e_0, e_1, e_2, \ldots, e_k$, then $exp(n, i)$ gives the g.n. of e_i. Further, note that if n is a g.n., then it consists in multiples of the first $len(n)$ primes (i.e. the primes from π_0 to $\pi_{len(n)-1}$).

We will now add another key p.r. function to our list:

8. There is a concatenation function such that (i) $m \star n$ is the g.n. of the expression that results from stringing together the expression with g.n. m followed by the expression with g.n. n, and (ii) this 'star' function is primitive recursive.[5]

Proof for (8) Suppose m is the g.n. of the expression '$\exists\mathsf{y}$', i.e. $m = 2^{11} \cdot 3^4$, and n is the g.n. of '$\mathsf{y} = \mathsf{0}$', i.e. $n = 2^4 \cdot 3^{13} \cdot 5^{19}$. Then we want $m \star n$ to deliver the g.n. of the concatenation of those two expressions, i.e. the g.n. of '$\exists\mathsf{y}\,\mathsf{y} = \mathsf{0}$', so we want $m \star n = 2^{11} \cdot 3^4 \cdot 5^4 \cdot 7^{13} \cdot 11^{19}$.

Look at the pattern of exponents here and generalize. Suppose therefore that m and n are Gödel numbers, and that $len(m) = j$ and $len(n) = k$. We want

[5] The use of the star symbol is traditional.

the function $m \star n$ to yield the value obtained by taking the first $j + k$ primes, raising the first j to powers that match the exponents (taken in order) of the j primes in the prime factorization of m and then raising the next k primes to powers that match the k exponents in the prime factorization of n. Then $m \star n$ will indeed yield the g.n. of the expression which results from stringing together the expression with g.n. m followed by the expression with g.n. n.

Now recall that bounded minimization keeps us in the sphere of the primitive recursive (Section 8.5(b)). It is then readily seen we can define a p.r. function $m \star n$ which applies to Gödel numbers in just the right way. We put[6]

$$\begin{aligned} m \star n = (\mu x \le B_{m,n})[(\forall i < \mathit{len}(m))\{\mathit{exp}(x,i) = \mathit{exp}(m,i)\} \land \\ (\forall i \le \mathit{len}(n))\{\mathit{exp}(x, i{+}\mathit{len}(m)) = \mathit{exp}(n,i)\}] \end{aligned}$$

where $B_{m,n}$ has to be a suitable primitive recursive function whose values keep the minimization operator finitely bounded. $B_{m,n} = \pi_{m+n}^{m+n}$ is certainly big enough to cover all eventualities. ⊠

(b) *The concatenation function in use* Here's an introductory pair of mini-examples of the star function at work. Suppose a is the g.n. of the wff $\exists \mathsf{x}\, \mathsf{x} = \mathsf{S0}$; then $\ulcorner \neg \urcorner \star a$ is the g.n. of $\neg\exists\mathsf{x}\,\mathsf{Sx} = \mathsf{0}$. Suppose b is the g.n. of '$\mathsf{S0} = \mathsf{0}$'. Then $\ulcorner (\urcorner \star a \star \ulcorner \rightarrow \urcorner \star b \star \ulcorner) \urcorner$ is the g.n. of $(\exists\mathsf{x}\,\mathsf{Sx} = \mathsf{0} \rightarrow \mathsf{S0} = \mathsf{0})$.

Note, by the way, that $((m \star n) \star o) = (m \star (n \star o))$, which is why we can suppress internal bracketing with the star function.

And here's a couple of results that we can easily prove now we have the star function to hand:

9. The function $\mathit{num}(n)$ whose value is the g.n. of the standard numeral for n is p.r.

10. The function $\mathit{diag}(n)$ is p.r.

Proof for (9) The standard numeral for Sn, for $n > 0$, is of the form 'S' followed by the standard numeral for n. So we have

$$\begin{aligned} &\mathit{num}(0) = \ulcorner \mathsf{0} \urcorner = 2^{19} \\ &\mathit{num}(Sx) = \ulcorner \mathsf{S} \urcorner \star \mathit{num}(x) = 2^{21} \star \mathit{num}(x) \end{aligned}$$

Hence *num* is primitive recursive. ⊠

Proof for (10) The *diag* function maps n, the g.n. of φ, to the g.n. of φ's diagonalization $\exists \mathsf{y}(\mathsf{y} = \ulcorner \varphi \urcorner \land \varphi)$. So we can put

$$\mathit{diag}(n) = \ulcorner \exists \mathsf{y}(\mathsf{y} = \urcorner \star \mathit{num}(n) \star \ulcorner \land \urcorner \star n \star \ulcorner) \urcorner$$

where *num* is as just defined. ⊠

(c) *Showing Termseq(m, n), Term(n), and Atom(n) are p.r.* Formal proofs in PA are sequences of wffs; wffs are built up from atomic wffs. And since the only

[6] A reality check: we are here doing *informal* mathematics. If we use the familiar quantifier and conjunction symbols, that is for brevity's sake. These aren't formal wffs! Cf Section 4.1.

primitive predicate of PA's language L_A is the identity relation, the atomic wffs are all expressions of the kind '$\sigma = \tau$' where σ, τ are *terms*. So let's focus first on the definition of terms.

A term is either '0', or a variable, or is built up from those using the function-symbols 'S', '+', '×' – to give us, for example, the complex term $(\mathsf{S0} \times (\mathsf{SS0} + \mathsf{x}))$ (see Section 4.3, (a)).

Now, let's say that a 'constructional history' for a term, or a *term-sequence*, is a sequence of expressions $\langle \tau_0, \tau_1, \ldots, \tau_n \rangle$ such that each expression τ_k in the sequence either (i) is '0'; or else (ii) is a variable; or else (iii) has the form $\mathsf{S}\tau_j$, where τ_j is an earlier expression in the sequence; or else (iv) has the form $(\tau_i + \tau_j)$, where τ_i and τ_j are earlier expressions in the sequence; or else (v) has the form $(\tau_i \times \tau_j)$, where τ_i and τ_j are earlier expressions. Since any well-formed term must have the right kind of constructional history, we can adopt as our official definition: *a term is an expression which is the last expression in some term-sequence.*

That last observation motivates our being interested in the particular numerical relation $Termseq(m, n)$, which holds when m is the super g.n. for a term-sequence, and n is the g.n. of the last expression in that term-sequence. And we can fairly readily prove the following results:

11. The property $Var(n)$ which holds when n is the g.n. of a variable is p.r.

12. The relation $Termseq(m, n)$ is p.r.

13. The property $Term(n)$ which holds when n is the g.n. of a term is p.r.

14. The property $Atom(n)$ which holds when n is the g.n. of an atomic wff is p.r.

Proof for (11) We just note that

$$Var(n) \leftrightarrow (\exists x \leq n)(n = 2^{2x})$$

For recall, the symbol code for a variable, on our scheme, always has the form $2x$, and the g.n. for a single expression is 2 to the power of the symbol code. The exponential function is p.r., and a definition by bounded quantification preserves primitive recursiveness, so Var is p.r. too.[7] ⊠

Proof for (12) We can define *Termseq* by something of the following shape:

$$Termseq(m, n) =_{\text{def}} exp(m, len(m) - 1) = n \land \\ (\forall k < len(m))\{\ldots\ exp(m, k)\ \ldots\}$$

[7] Of course, this simple definition of *Var* depends on our particular coding scheme, and on our treating all variables as single 'symbols' even if in fact built up from a more primitive alphabet by using primes or whatever: alternative methodical schemes will generate other p.r. functions.

The *first* conjunct on the right ensures that, in the sequence with super g.n. m, the last expression indeed has g.n. n. We now need to fill out the curly brackets in the *second* conjunct in a way that reflects the fact for each $k < len(m)$, $exp(m,k)$ – the g.n. of τ_k in our putative term-sequence – is the g.n. of an expression satisfying one of the five defining conditions for belonging to a term-sequence. So, in those curly brackets, we therefore need to say that $exp(m,k)$ is either

(i) the g.n. of '0', so $exp(m,k) = \ulcorner 0 \urcorner$;
or (ii) the g.n. of a variable, so $Var(exp(m,k))$;
or (iii) the g.n. of '$\mathsf{S}\tau_j$' where τ_j occurs earlier in the term sequence, so
$(\exists j < k)(exp(m,k) = \ulcorner \mathsf{S} \urcorner \star exp(m,j))$;
or (iv) the g.n. of '$(\tau_i + \tau_j)$' where τ_i and τ_j occur earlier, so
$(\exists i < k)(\exists j < k)(exp(m,k) = \ulcorner (\urcorner \star exp(m,i) \star \ulcorner + \urcorner \star exp(m,j) \star \ulcorner) \urcorner)$;
or (v) the g.n. of '$(\tau_i \times \tau_j)$' where τ_i and τ_j occur earlier, so
$(\exists i < k)(\exists j < k)(exp(m,k) = \ulcorner (\urcorner \star exp(m,i) \star \ulcorner \times \urcorner \star exp(m,j) \star \ulcorner) \urcorner)$

All the clauses are p.r. conditions, so their disjunction is a p.r. condition, so *Termseq* is p.r., as we wanted to show. ⊠

Proof for (13) Since a term has to be the final member of some term sequence, we can give the following definition:

$$Term(n) =_{\text{def}} (\exists x \leq B_n)\, Termseq(x,n)$$

where B_n is a suitably large bound. Given a term with g.n. n, and hence with $l = len(n)$ symbols, its term-sequence will be at most l long: so the super g.n. of any term-sequence constructing it must be less than $B_n = (\pi_l^n)^l$. ⊠

Proof for (14) In PA, the only atomic wffs are expressions of the kind $\tau_1 = \tau_2$. So put

$$Atom(n) =_{\text{def}}$$
$$(\exists x \leq n)(\exists y \leq n)[Term(x) \wedge Term(y) \wedge n = (x \star \ulcorner = \urcorner \star y)]$$

It is then immediate that $Atom(n)$ is p.r. ⊠

(d) *Showing Formseq(m,n), Wff(n), and Sent(n) are p.r.* We can now proceed to define the idea of a *formula-sequence* analogously to the idea of a term-sequence. So: a formula-sequence is a sequence of expressions $\langle \varphi_0, \varphi_1, \ldots, \varphi_n \rangle$ such that each φ_k in the sequence is either an atomic wff or built from previous wffs in the sequence by using a connective or by prefixing any quantifier.[8] And we'll define $Formseq(m,n)$ to be true when m is the super g.n. of a formula-sequence, and n is the g.n. of the last expression in that sequence. This too is primitive recursive. And to show that, we can obviously use exactly the same

[8] A technical remark: things go easiest here if we give L_A a liberal syntax which allows wffs with redundant quantifiers which don't bind any variables in their scope.

general strategy as in our proof that *Termseq* is p.r. We won't spell out the boring details here (we can explore how to do this in the exercises).

Since any wff by definition will be the final formula in a formula sequence, then – again by choosing a new suitable bound B_n – we can define the p.r. property of numbering a wff:

$$\mathit{Wff}(n) =_{\mathrm{def}} (\exists x \leq B_n)\mathit{Formseq}(x, n).$$

And with a few more wrinkles, we can also give a similar construction to show that *Sent* is p.r., where $\mathit{Sent}(n)$ holds just when n is the g.n. of a sentence, i.e. a closed wff with no free variables. Again we won't give the tedious details here.

(e) *Showing* $\mathit{Prfseq}(m, n)$ *is p.r.* And now the end is in sight: we can now outline a proof that *Prfseq* is primitive recursive as claimed.

The details, though, will depend heavily on the type of proof system we are dealing with. But remember, in order to keep things particularly simple, we're supposing that we have adopted a very old-fashioned linear proof-system for PA (which therefore doesn't allow temporary suppositions and sub-proofs). In this kind of system there are logical axioms – e.g. instances of the schemas '$((\varphi \wedge \psi) \rightarrow \varphi)$' and '$(\forall\xi\varphi(\xi) \rightarrow \varphi(\tau))$' – and we can manage with just one rule of inference, modus ponens.[9] We can then define a PA proof as a sequence of wffs, each one of which is either an axiom (a PA axiom or a logical axiom), or follows from previous wffs in the sequence by modus ponens.

Now, the relation $\mathit{Modusponens}(m, n, o)$ – which holds when the wff with g.n. o follows from the wffs with g.n. m and n by modus ponens – is obviously p.r., for it is immediate that $\mathit{Modusponens}(m, n, o)$ just when

$$m = \ulcorner(\urcorner \star n \star \ulcorner\rightarrow\urcorner \star o \star \ulcorner)\urcorner \ \wedge \ \mathit{Wff}(n) \ \wedge \ \mathit{Wff}(o).$$

It's trickier to prove the property $\mathit{Axiom}(n)$ is p.r., where $\mathit{Axiom}(n)$ holds just when n numbers a wff which is an axiom (whether a specifically arithmetical axiom of PA or a logical axiom). That's because, in order to deal with the instances of the logical axiom schemas for quantification, we'll have to arithmetize facts about which variables are free in which wffs. And in order to deal with the arithmetical induction axioms, we'll need to be able to handle all the numbers that code for the universal closures of instances of the induction axiom. We won't go into details (again we'll explore in the exercises how we might go about the task).

But if we can assume that $\mathit{Axiom}(n)$ is p.r., it is easy to get to our final target result: the relation $\mathit{Prfseq}(m, n)$ is indeed p.r.:

Proof for the main result We echo the pattern of our definition of *Termseq*:

> $\mathit{Prfseq}(m, n) =_{\mathrm{def}} \mathit{exp}(m, \mathit{len}(m) - 1) = n$ and $\mathit{Sent}(n)$ and also $\forall k < \mathit{len}(m)$, $\mathit{exp}(m, k)$ – i.e. the g.n. of the k-th expression of the sequence – is either

[9] As in the system *QS* of (Hunter, 1971, pp.167-168).

(i) the g.n. of an axiom, so $Axiom(exp(m, k))$;

or (ii) the g.n. of a wff that follows from two earlier wffs by MP, so $(\exists i \leq k)(\exists j \leq k)Modusponens(exp(m, i), exp(m, j), exp(m, k))$;

Which is p.r. since it is built up in p.r. ways from p.r. components. ⊠

(f) *Provability* So much, then, for our promised outline of the stages of a proof that $Prfseq(m, n)$ is primitive recursive. One comment before leaving this, however. $Prfseq(m, n)$ holds when m is the super g.n. of a proof of the wff with g.n. n: so $Prov(n) =_{\text{def}} \exists v\, Prfseq(v, n)$ holds when the wff with g.n. n is provable. Note, though, that we *can't* read off from n some upper bound on the length of possible proofs for the wff with g.n. n. So we *can't* just define the provability property by some *bounded* quantification of the kind $(\exists v \leq B)Prfseq(v, n)$. If we could, then the provability probability would be p.r.: but it isn't – as we will show in Section 15.6.

13 PA is incomplete

The pieces we need are at long last all in place. So in this chapter we can finally give Gödel's quite beautiful construction which delivers a true-but-unprovable sentence for PA. Then in the next chapter, we show how this result can be generalized to prove that PA – and any other formal arithmetic satisfying very modest constraints – is not only incomplete but incompletable. Our discussion in these two chapters fairly closely follows Gödel's own treatment in 1931. In Chapter 15 we'll dig a bit deeper and give a slightly more modern take on Gödel's proof: but I think it is well worth taking the historical route first.

13.1 Constructing *G*

We've already trailed the headline news: Gödel constructs a PA sentence G that 'says', when read in the light of the Gödel coding, 'I am unprovable in PA'. But how does he do this?

First, two reminders. Recall (from Section 12.5):

> The diagonalization of φ is $\exists \mathsf{y}(\mathsf{y} = \ulcorner\varphi\urcorner \wedge \varphi)$, where '$\ulcorner\varphi\urcorner$' here stands in for the numeral for φ's g.n.

If φ is a wff $\varphi(\mathsf{y})$ with 'y' as its sole free variable, then φ's diagonalization is trivially equivalent to $\varphi(\ulcorner\varphi\urcorner)$.

Next, recall another definition (from Section 12.6):

> *Gld*(m, n) is true when m is the super g.n. for a PA proof of the diagonalization of the wff with g.n. n.

As we saw, *Gld* can be simply defined by composition from the p.r. relation *Prfseq* and the p.r. function *diag*, so is itself primitive recursive. Hence by Theorem 14 (Section 11.2), there must be an open wff of Q, and hence of PA, which *expresses* it. Indeed, by the stronger Theorem 16 (Section 11.6), there must be an open wff of Q, and hence of PA, which *captures* it as a function.

More precisely, there will be a wff $\mathsf{Gdl}(\mathsf{x}, \mathsf{y})$ built up by a chain of definitions that recapitulate the relation's – or rather, its characteristic functions's – p.r. definition. This wff, with definitions unpacked, will be Σ_1. And it will both express *Gld* and capture it as a function.

Consider, then, the PA wff

$$\forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \mathsf{y})$$

Abbreviate this wff as U – or as $\mathsf{U}(\mathsf{y})$ when we want to stress that it contains just 'y' free. And now diagonalize U:

$$\exists \mathsf{y}(\mathsf{y} = \ulcorner \mathsf{U} \urcorner \wedge \mathsf{U}(\mathsf{y}))$$

This is the 'Gödel sentence' for PA which we've been aiming to construct: we'll abbreviate it G.

We immediately have the trivial logical equivalence of G with $\mathsf{U}(\ulcorner \mathsf{U} \urcorner)$. Or unpacking that a bit,

G is equivalent to $\forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \ulcorner \mathsf{U} \urcorner)$.

Consider, then, what it takes for G to be true, in the light of the fact that the formal predicate Gdl expresses the numerical relation *Gld*. By our equivalence, G is true if and only if there is no number m such that $Gld(m, \ulcorner \mathsf{U} \urcorner)$. That is to say, given the definition of *Gld*, G is true if and only if there is no number m such that m is the code number for a PA proof of the diagonalization of the wff with g.n. $\ulcorner \mathsf{U} \urcorner$. But by construction, the wff with g.n. $\ulcorner \mathsf{U} \urcorner$ is the wff U; and its diagonalization is G. So, G is true if and only if there is no number m such that m is the code number for a PA proof of G. But if G is provable, some number would be the code number of a proof of it. Hence G is true if and only if it is unprovable in PA. Wonderful!

13.2 Interpreting *G*

It is often claimed that G is not merely true if and only if unprovable in PA, but it actually '*says*' that it is unprovable in PA: indeed, for vividness, we have occasionally put it that way. But we must be *very* cautious here!

The wff G is *just another sentence of* PA*'s language* L_A, the language of basic arithmetic. It is an enormously long wff involving the first-order quantifiers, the connectives, the identity symbol, and 'S', '+' and '×', which all have the standard interpretation built into L_A.[1] And on that interpretation, G is strictly speaking a complex claim about the results of ordinary arithmetical operations, no more and no less.

To emphasize the point, the semantics built into L_A tells us that the various terms in G (numerals and numerical function expressions) have *numbers* as values. In other words, their values aren't non-numerical items like, e.g., linguistic expressions. Hence, in particular, G does not straightforwardly refer to itself.

So what is going on if we summarily claim that G 'says' it is unprovable, when read in the light of Gödel coding? Note that this is *not* to invoke some

[1] The wff G embeds the standard numeral for U(y)'s Gödel number: and given our particular numbering scheme this number will be *huge*. So writing out G in *pure* L_A *without abbreviations* would not be a practical possibility.

How significant is this fact? Well, the situation is actually not at all unusual in mathematics. Given our limited cognitive powers, we need to use chains of abbreviations all the time, and we very often work with propositions whose official definitional unpacking into the relevant 'basic' terms would be far too long to grasp. This general fact certainly isn't without its interest and its problematic aspects: for related discussion, see e.g. Isles (1992). However, the general fact cuts right across mathematics. So we aren't going to worry for the moment about its specific application to the Gödel sentence G.

radical re-interpretation of its symbols (for doing *that* would just make the claim quite trivial: if we are allowed radical re-interpretations – like spies choosing to borrow ordinary words for use in a secret code – then any string of symbols can be made to mean anything). No, it is because the symbols are *still* being given their *standard* interpretation that we can recognize that Gdl (when unpacked) will express *Gld*, given the background framework of Gödel numbering which is involved in the definition of the relation *Gld*. And therefore, knowing about the Gödel numbering, we can recognize from its construction that G will hold when no number m is such that $Gld(m, \ulcorner \mathsf{U} \urcorner)$. So we can then immediately see, without further argument, that G is constructed in such as way as to make it true just when it is unprovable in PA. *That* is the limited sense in which it 'says' – with heavy scare quotes – that it is unprovable.

13.3 *G* is undecidable in PA: the semantic argument

The argument that G is 'formally undecidable' now runs along the lines we sketched right back in Section 1.2.

Assume that the axioms of PA are true on the standard interpretation built into PA's language L_A (which they are, of course). We know that PA's logic – i.e. some version of classical first-order logic – is necessarily truth-preserving. Hence all its theorems are true: PA proves no falsehoods. But if G (which is true if and only if it is unprovable) can be proved in PA, the theory *does* prove a false theorem. Contradiction! Hence, G is not provable in PA. Hence G is true on the standard interpretation. So ¬G is false. Hence ¬G cannot be proved in PA either. In Gödel's words, G is a 'formally undecidable' wff of PA. So that gives us

> **Theorem 18** *If* PA *has true axioms, then there is a closed wff* φ *of* L_A *such that neither* $\mathsf{PA} \vdash \varphi$ *nor* $\mathsf{PA} \vdash \neg\varphi$.

And if we are happy with the semantic assumption that PA's axioms are true on interpretation, the argument for incompleteness is as simple as that, once we have constructed G.[2]

Now, for reasons that will become clearer when we consider Hilbert's programme and related background in a later Interlude, it was very important to Gödel that incompleteness can be proved *without* supposing that PA is sound: as he puts it, 'purely formal and much weaker assumptions' suffice. However, the further argument that shows this is a bit tricksier (especially when we move to Rosser's enhanced version of the argument in Section 14.4, which is needed to get the best non-semantic analogue for Theorem 18).

So don't lose sight of our much simpler 'semantic' argument for incompleteness.

[2] Note, then, that Theorem 18 only requires us to be able to construct G via a wff which *expresses Gdl*. We haven't yet invoked the stronger claim that the numerical relation *Gdl* can be *captured* in PA.

13.4 ‘G is of Goldbach type’

Before showing how ‘weaker assumptions suffice’, let’s pause to remark that the unprovable Gödel sentence G, while in one way horribly complex (when spelt out in unabbreviated L_A), is in another way very simple. For the open wff $\mathsf{Gdl(x,y)}$ is Σ_1, and it expresses a p.r. relation. So $\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$ is also Σ_1, and expresses a p.r. property. So its negation $\neg\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$ is Π_1: and that also expresses a p.r. property, since the negation of a p.r. property is still p.r. (see Section 9.7). Hence $\forall\mathsf{x}\neg\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$ is Π_1 too, and is a universal generalization about a p.r. property. Its logical equivalent G is therefore also Π_1.

Which means that the Gödel sentence G is on a par with e.g. a natural formal statement of Goldbach’s conjecture, which is another Π_1 wff and so is also equivalent to a universal generalization about a p.r. property.[3]

So rather than talking of Π_1 wffs, then, we can perhaps make things more vivid by talking of a proposition being of *Goldbach type*, meaning that it is equivalent to universal generalization about a p.r. property or relation. Then we can strengthen our statement of Theorem 18, to give us

> **Theorem 18*** *If* PA *has true axioms, then there is a closed* L_A*-wff* φ *of Goldbach type such that neither* $\mathsf{PA}\vdash\varphi$ *nor* $\mathsf{PA}\vdash\neg\varphi$.

13.5 G is unprovable in PA: the syntactic argument

We’ll show again that PA can’t prove G, but this time *without* assuming PA’s soundness: we’ll just make an assumption of consistency. So we’ll prove

A. If PA is consistent, $\mathsf{PA}\nvdash\mathsf{G}$.

Proof Suppose G *is* provable in PA. If G has a proof, then there is some super g.n. m that codes its proof. But by definition, G is the diagonalization of the wff U. Hence, by definition, $Gld(m,\ulcorner\mathsf{U}\urcorner)$.

Now we use the fact that Gdl not just expresses but *captures* the relation Gld. That implies (i) $\mathsf{PA}\vdash\mathsf{Gdl}(\mathsf{m},\ulcorner\mathsf{U}\urcorner)$.

[3] See Section 7.5 for the demonstration that Goldbach’s conjecture is Π_1. It is therefore the universal generalization of a Δ_0 wff. And any Δ_0 wff expresses a p.r. property or relation. For evidently an atomic Δ_0 wff expresses a p.r. property or relation, and we know that applying propositional connectives and/or bounded quantifiers keeps us within the realms of the primitive recursive – see Section 9.7, (c). Hence Goldbach’s conjecture is a universal generalization about a p.r. property.

Generalizing this argument shows that *any* Π_1 sentence is equivalent to a universal generalization about a p.r. property or relation. And the converse holds too. Suppose we want to express a universal generalization about some p.r. property or relation. Then we can always use a Π_1 wff to do the job. To take the simplest case, suppose we want to say that every number has the p.r. monadic property P. Since P is p.r., its characteristic function c_P will be p.r., so c_P is expressible by a Σ_1 wff (by Theorem 14 again), and so also expressible by a Π_1 wff $\varphi(\mathsf{x},\mathsf{y})$ (by the Σ_1/Π_1 lemma of Section 11.1). So then the Π_1 wff $\varphi(\mathsf{x},\mathsf{0})$ will express P and the Π_1 wff $\forall\mathsf{x}\varphi(\mathsf{x},\mathsf{0})$ will express the claim that every number is P.

But since G is logically equivalent to $\forall\mathsf{x}\neg\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$, the assumption that G is provable comes to this: $\mathsf{PA} \vdash \forall\mathsf{x}\neg\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$. The universal quantification here entails any instance. Hence (ii) $\mathsf{PA} \vdash \neg\mathsf{Gdl}(\mathsf{m},\ulcorner\mathsf{U}\urcorner)$.

So, combining (i) and (ii), the assumption that G is provable entails that PA is inconsistent. Hence, if PA is consistent, there can be no PA proof of G. ⊠

13.6 ω-incompleteness, ω-inconsistency

(a) G is unprovable in PA. So no number is the super g.n. of a proof of G. That is to say, no number numbers a proof of the diagonalization of U. That is to say, for any particular m, it *isn't* the case that $Gld(m,\ulcorner\mathsf{U}\urcorner)$. So, again by the fact that Gdl captures Gld we have

1. For each m, $\mathsf{PA} \vdash \neg\mathsf{Gdl}(\mathsf{m},\ulcorner\mathsf{U}\urcorner)$.

2. But, assuming consistency, $\mathsf{PA} \nvdash \mathsf{G}$ – i.e., $\mathsf{PA} \nvdash \forall\mathsf{x}\neg\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$.

Now for a standard definition:

> An arithmetic theory T is *ω-incomplete* if, for some open wff $\varphi(\mathsf{x})$, T can prove each $\varphi(\mathsf{m})$ but T can't go on to prove $\forall\mathsf{x}\varphi(\mathsf{x})$.

('ω' is the logicians' label for the natural numbers taken in their natural order.) So the case where $\varphi(\mathsf{x}) =_{\mathrm{def}} \neg\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$ shows that PA is ω-incomplete.

In Section 6.3, we noted that Q exhibits a radical case of what we are now calling ω-incompleteness: although it can prove case-by-case all true equations involving numerals, it can't prove many of their universal generalizations. For a simple example, put $\mathsf{Kx} =_{\mathrm{def}} (\mathsf{0} + \mathsf{x} = \mathsf{x})$; then for every n, $\mathsf{Q} \vdash \mathsf{Kn}$; but we don't have $\mathsf{Q} \vdash \forall\mathsf{xKx}$. In moving from Q to PA by adding the induction axioms, we vastly increase our ability to prove generalizations. But we now see that some ω-incompleteness must remain even in PA.

(b) Here's another standard definition:

> An arithmetic theory T is *ω-inconsistent* if, for some open wff $\varphi(\mathsf{x})$, $T \vdash \exists\mathsf{x}\varphi(\mathsf{x})$, yet for each number m we have $T \vdash \neg\varphi(\mathsf{m})$.

So compare and contrast. Suppose T can prove $\psi(\mathsf{m})$ for each m. T is ω-incomplete if it can't prove something we'd *like* it to prove, namely $\forall\mathsf{x}\psi(\mathsf{x})$. While – putting ψ for $\neg\varphi$ in our definition – T is ω-inconsistent if it can actually prove the *negation* of what we'd like it to prove, i.e. it can prove $\neg\forall\mathsf{x}\psi(\mathsf{x})$, i.e. $\exists\mathsf{x}\neg\psi(\mathsf{x})$.

Note that ω-inconsistency, like ordinary inconsistency, is a syntactically defined property: it is characterized in terms of what wffs can be proved, not in terms of what they mean. Note too that ω-consistency – defined of course as not being ω-inconsistent! – implies plain consistency. That's because T's being

ω-consistent is a matter of its *not* being able to prove a certain combination of wffs, which entails that T can't be inconsistent and prove *all* wffs.

Now, ω-incompleteness is regrettable; but ω-inconsistency in a theory of arithmetic is a Very Bad Thing (not as bad as outright inconsistency, maybe, but still bad enough). For suppose that the axioms of an arithmetic theory T are indeed true when given a *normal* interpretation – by which we mean an interpretation whose domain comprises just zero and its successors, on which T's standard numerals are correctly assigned to the corresponding natural numbers, and logical apparatus is treated as usual. Assuming T has a sound logic, T's theorems will then all be true on this interpretation. So now suppose that, for some $\varphi(\mathsf{x})$, T does prove each of $\neg\varphi(\mathsf{0}), \neg\varphi(\mathsf{1}), \ldots, \neg\varphi(\mathsf{n}), \ldots$. By hypothesis, these theorems will then be true on the given normal interpretation; so this means that every natural number must satisfy $\neg\varphi(\mathsf{x})$. Hence there is no object in the domain left over to satisfy $\varphi(\mathsf{x})$. So $\exists\mathsf{x}\varphi(\mathsf{x})$ will be have to be false on this normal interpretation. Therefore it can't be a theorem. Hence, contraposing, if T is ω-inconsistent – i.e. each of $\neg\varphi(\mathsf{n})$ is a theorem and yet $\exists\mathsf{x}\varphi(\mathsf{x})$ is a theorem too – then T's axioms can't all be true on a normal arithmetic interpretation.

Given we want arithmetics to have axioms which *are* all true on a normal interpretation, we must want ω-consistent arithmetics. (And given we think PA is sound on its normal interpretation, we are committed to thinking that it *is* ω-consistent.)

13.7 $\neg G$ is unprovable in PA: the syntactic argument

We'll now show again that PA can't prove the negation G, but this time *without* assuming PA's soundness: we'll just make the syntactic assumption of ω-consistency.

B. If PA is ω-consistent, PA $\nvdash \neg\mathsf{G}$.

Proof Suppose that PA is ω-consistent but $\neg\mathsf{G}$ is provable in PA. That's equivalent to assuming (i) PA $\vdash \exists\mathsf{x}\mathsf{Gdl}(\mathsf{x}, \ulcorner\mathsf{U}\urcorner)$.

But if PA is ω-consistent, it is consistent. So if $\neg\mathsf{G}$ is provable, G is *not* provable. Hence for any m, m cannot code for a proof of G. But G is (again!) the wff you get by diagonalizing U. Therefore, by definition, our assumptions imply not-$Gld(m, \ulcorner\mathsf{U}\urcorner)$, for each m.

So, by the requirement that Gdl captures *Gld*, we have (ii) PA $\vdash \neg\mathsf{Gdl}(\mathsf{m}, \ulcorner\mathsf{U}\urcorner)$ for each m.

But (i) and (ii) together make PA ω-inconsistent after all, contrary to hypothesis. Hence, if PA is ω-consistent, $\neg\mathsf{G}$ is unprovable.[4] ⊠

[4] This is a meaty footnote for enthusiasts, embroidering our result in two related ways, mainly to link up with presentations in other books. But do skip it on a first reading!

The first point to note is that the combination of (i) and (ii) is also ruled out by a weaker assumption than PA's ω-consistency.

That means, by the way, that the theory $\mathsf{PA} + \neg\mathsf{G}$ (which you get by adding $\neg\mathsf{G}$ as a new axiom to Peano Arithmetic) must be ω-inconsistent; yet this expanded theory is consistent, by (A) from Section 13.5. Which confirms that ω-consistency is a stronger requirement than mere consistency.

13.8 Putting things together

Putting results (A) and (B) together with the fact that G is of Goldbach type, we therefore have the following crucial result:

> **Theorem 19** *There is an L_A-sentence φ of Goldbach type such that, if* PA *is consistent then* $\mathsf{PA} \nvdash \varphi$*; and if* PA *is ω-consistent then* $\mathsf{PA} \nvdash \neg\varphi$*.*

And we should immediately stress two points about all this. First, our existence claim is proved 'constructively'. That is to say, we haven't just given an abstract existence-proof that there must be a formally undecidable sentence (as in Section 5.3): we have now given a *recipe* for actually constructing a 'formally undecidable' Gödel sentence φ.

Second, we have in fact done more: we have given a recipe for producing arbitrarily many distinct undecidable sentences! Suppose we just fiddle around with the details of our Gödel-numbering scheme. Then, as we change the scheme, the details of the corresponding p.r. relation *Gld* will change and the corresponding capturing wff Gld will have to change too – and then we'll get new undecidable sentences modelled on G.

We'll say a theory is *1-consistent* if it is ω-consistent, perhaps not across the board, but at least at the level of Σ_1 wffs. More carefully: an arithmetic theory T is 1-consistent if there is no open Σ_1 wff $\varphi(\mathsf{x})$ such that $T \vdash \exists\mathsf{x}\varphi(\mathsf{x})$ yet for each m $T \vdash \neg\varphi(\mathsf{m})$. But $\mathsf{Gdl}(\mathsf{x},\mathsf{y})$ is Σ_1, so $\exists\mathsf{x}\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$ is Σ_1. Hence, so long as PA is 1-consistent, we can't have (i) and (ii) true together.

We can therefore strengthen our result: if PA is 1-consistent, $\mathsf{PA} \nvdash \neg\mathsf{G}$.

It is also worth noting that, if T contains Q, 1-consistency is equivalent to Σ_1-soundness (we defined that in Section 7.4: T is Σ_1-sound if, whenever φ is Σ_1 and $T \vdash \varphi$, then φ is true).

Proof sketch Note first that if T is inconsistent, it proves any Σ_1-wff, and some of these like $\exists\mathsf{x}(\mathsf{Sx} = \mathsf{0})$ will be false, so T can't be Σ_1-sound. So, contraposing, Σ_1-soundness implies consistency. Now suppose that T *is* Σ_1-sound. Then if $T \vdash \exists\mathsf{x}\psi(\mathsf{x})$, with $\psi(\mathsf{x})$ a Δ_0 wff, $\exists\mathsf{x}\psi(\mathsf{x})$ is true. So some $\psi(\overline{\mathsf{m}})$ is true. But $\psi(\overline{\mathsf{m}})$, being Δ_0, is therefore also Σ_1; and so it is provable in T (since T contains Q, and Q is Σ_1-complete). Hence by T's consistency, $\neg\psi(\overline{\mathsf{m}})$ *isn't* provable, and therefore T is 1-consistent. Conversely, suppose T is not Σ_1-sound, i.e. proves some $\exists\mathsf{x}\psi(\mathsf{x})$ which is false (so $\forall\mathsf{x}\neg\psi(\mathsf{x})$ is true). Then every $\neg\psi(\overline{\mathsf{m}})$ is true, and – being Σ_1 – is provable in T. Which makes T 1-inconsistent. ☒

Which gives us another version of our strengthened result: if PA is Σ_1-sound, then $\mathsf{PA} \nvdash \neg\mathsf{G}$.

14 Gödel's First Theorem

Back in Chapter 6, we introduced the weak arithmetic Q, and soon saw that it is boringly incomplete. Then in Chapter 8 we introduced the much stronger first-order theory PA, and remarked that we couldn't in the same easy way show that it fails to decide some elementary arithmetical claims. However, in the last chapter it has turned out that PA is incomplete too.

Still, that result in itself isn't yet hugely exciting, even it is a bit surprising. After all, just saying that a particular theory T is incomplete leaves wide open the possibility that we can patch things up by adding an axiom or two more, to get a complete theory T^+. As we said at the very outset, the real force of Gödel's proof is that it illustrates a *general* method which can be applied to *any* theory satisfying modest conditions to show that it is incomplete. And this reveals that a theory like PA is not only incomplete but in a good sense *incompletable.*

The present chapter starts by explaining this quite crucial point. Then we note an important strengthening of Gödel's proof due to Rosser. The third main theme of the chapter involves using a trick due to William Craig to widen the import of our incompleteness results.

14.1 Generalizing the semantic argument

In Section 13.3, we showed that PA is incomplete on the semantic assumption that its axioms are true (and its logic is truth-preserving). In this section, we are going to extend this style of argument to other theories.

We begin with an important definition. We said in Section 3.3 that a theory T is an axiomatized formal theory if it is (a) effectively decidable what counts as a T-wff and what counts as a T-sentence, (b) it is effectively decidable which wffs are T-axioms, and (c) T uses a proof-system such that it is effectively decidable whether an array of T-wffs counts as a well-constructed derivation. We'll now say that a theory T is *p.r. axiomatized* if (a$'$) the numerical properties of being the g.n. of a T-wff and T-sentence are primitive recursive, and similarly (b$'$) the numerical property of being the g.n. of an axiom is p.r., and likewise (c$'$) the numerical property of being the super g.n. of a properly constructed proof is p.r. too.[1]

[1] Note, by the way, our remark at the end of Section 12.1, where we explained why a numerical property like being the g.n. a T-wff will be p.r. on any acceptable numbering scheme if it is p.r. on our default Gödel-style scheme. So the question whether a theory is p.r. axiomatized is not embarrassingly relative to our particular numbering scheme.

Now looking again at our argument for Theorem 18, we can see that one essential fact underpinning our 'semantic argument' for the incompleteness of PA is:

1. There is an open wff Gdl which *expresses* the relation *Gld* – in other words, $\mathsf{Gdl}(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is true just so long as $Gld(m, n)$, i.e. just so long as m numbers a proof in PA of the diagonalization of the wff numbered by n.

But, let's not forget, we also need another fact:

2. We have quantification available.[2]

Quantification enables us to form the wff G which is true if and only if G is unprovable in PA. And then, assuming PA is a sound theory, we can show that neither G nor $\neg$G is provable.

Underpinning (1), we have in turn the facts that

3. The relation *Gld* is primitive recursive (see Sections 12.6, 12.7).
4. PA's language L_A can express all p.r. functions (see Section 11.4).

But *Gld* can be defined in terms of *Prfseq* (and the trivially p.r. *diag* function). And reviewing our proof that *Prfseq* is p.r. in Section 12.7, we see that it pivots around the fact that

5. PA is p.r. axiomatized.

For it assumes that *sent* and *axiom* are p.r., and that the relations that holds between the codes for the input and output of inference rules are p.r. too.

So, now to generalize. Suppose that we are dealing with any other theory T such that

G1. T is p.r. axiomatized;

G2. T's language includes L_A,

where we'll say T's language includes L_A if (i) every L_A wff is also a wff of T, perhaps allowing for some definitional extensions of T's original language,[3] and (ii) the copies of L_A wffs in T are true when their originals are true. Then (G1) gives us the analogue of (3). And (G2) gives us the analogues of (4) and (2).

[2] That we need quantification here is important! In Section 10.5, we outlined the construction of the theory $\mathsf{PRA_0}$ which expresses all p.r. functions, so it can express the version of *Gdl* defined for that theory, so the analogue of (1) holds. However, the absence of quantifiers in the language of $\mathsf{PRA_0}$ blocks our going on to form a Gödel sentence for that weak theory. Which is why the theory can, as we claimed, be negation-complete.

[3] There is a small wrinkle here. Suppose T is some non-arithmetic theory like set theory. To extend T's language to subsume L_A, we'll need to allow the definition of new *function* expressions for the successor, addition and multiplication functions, even if the original language lacks function expressions. But there are standard ways of handling this: see e.g. (Mendelson, 1997, pp. 103–105).

So there will be an open wff Gdl_T which expresses the relation Gld_T, i.e. the relation which holds just so long as m numbers a T-proof of the diagonalization of the wff numbered by n. And we can then form a corresponding new Gödel sentence G_T which 'says' that it is unprovable in T.

Of course, when we move to consider a different theory T, the set of axioms and/or the set of rules of inference will change (and if T involves new symbols, then the scheme for Gödel-numbering will need to be extended). So the details of the corresponding relation $Gld_T(m, n)$ will change too. Hence Gdl_T will be new, and likewise G_T. But still, we can construct this new Gödel sentence along just the same lines as before, needing to use only basic arithmetical vocabulary.

And now exactly the same easy argument as we used to establish Theorem 18 can be used to show

> **Theorem 20** *If a theory T, whose language includes L_A, is p.r. axiomatized, and is sound (i.e. it has true axioms and a truth-preserving logic), then there is an L_A sentence φ of Goldbach type such that $T \nvdash \varphi$ and $T \nvdash \neg\varphi$.*

14.2 Incompletability – a first look

Suppose T is a p.r. axiomatized, sound theory, whose language includes L_A; and suppose φ is one of the undecided sentences such that neither $T \vdash \varphi$ nor $T \vdash \neg\varphi$. One of φ, $\neg\varphi$ is true. Consider the result of adding the true one to T as a new axiom. The new expanded theory T^+ is still a p.r. axiomatized, sound theory, whose language includes L_A. So, although T^+ by construction does now decide φ the right way, it is still incomplete.

And so it goes. Throw as many new true axioms into T as you like, even augment the truth-preserving deductive apparatus, and the result will still be incomplete – unless it ceases to be p.r. axiomatized. In short, T is not just incomplete, but it is incompletable (if we still want a sound, p.r. axiomatized theory).

14.3 The First Theorem, at last

(a) So far, so good. Now we turn to generalizing the syntactic argument for incompleteness. Looking at our proof for Theorem 19, we can see that the essential facts underpinning *this* incompleteness proof are:

1′. There is an open wff Gdl which *captures* the relation Gld.

2. We have quantification available (as before).

And underpinning (1′), we have the facts that

3. The relation Gld is primitive recursive (as before).

4′. PA can capture all p.r. functions (see Section 11.6).

But, to repeat, (3) depends essentially on the fact that

5. PA is p.r. axiomatized

And now recall another definition: a theory is p.r. adequate if it captures each p.r. function as a function – where capturing as a function is defined using the universal quantifier (see Sections 10.2, 10.5). So we can in fact can wrap up conditions (2) and (4′) into one:

6. PA is p.r. adequate.

Suppose then that we are dealing with any other theory T such that

G1. T is p.r. axiomatized.

G2′. T is p.r. adequate.

Then (G1) gives us the analogue of (3), i.e. there will be a p.r. relation Gld_T which holds just so long as m numbers a T-proof of the diagonalization of the wff numbered by n. And (G2′) gives us some arithmetic language involving quantification and gives us the analogue of (4′). Hence there will be an open wff Gdl_T which captures the relation Gld_T. Hence, using the same construction, we can again form a corresponding new Gödel sentence G_T of Goldbach type; then the line of argument will go exactly as before. Therefore

> **Theorem 21** *If T is a p.r. adequate, p.r. axiomatized theory, then there is an arithmetic T-sentence φ of Goldbach type such that, if T is consistent then $T \nvdash \varphi$, and if T is ω-consistent then $T \nvdash \neg\varphi$.*

And it is this very general syntactic version of the incompleteness result which probably has as much historical right as any to be called Gödel's First Incompleteness Theorem.[4] So we get there at last!

(b) Another definition. Let's now call a theory simply *nice* if it is consistent, p.r. axiomatized, and includes Q (i.e. can prove at least everything that Q can prove).[5] Since we know – as Gödel didn't in 1931 – that even Q is p.r. adequate, another version of the First Theorem is:

> **Theorem 21*** *If T is any nice theory then there is an arithmetic T-sentence φ of Goldbach type such that neither $T \vdash \varphi$ nor (assuming T is ω-consistent) $T \vdash \neg\varphi$.*

[4] Not that the Theorem is stated quite like that in the original paper. We have put together Theorem VI and VIII of Gödel (1931), together with the generalizing gloss at the end of Section 2 of the paper.

[5] There doesn't seem to be a standard bit of jargon for labelling such theories: so 'nice' is just my own informal snappy shorthand.

Strictly speaking, these two versions are not *quite* equivalent because there are actually some alternative very weak p.r. adequate arithmetics which neither contain nor are contained in Q.[6] But the little differences won't matter to us, and in fact – although it isn't so historically accurate – we'll henceforth take Theorem 21* as our preferred standard form of the First Theorem.[7]

This obviously gives us another incompletability result. Suppose we beef up some nice ω-consistent theory T by adding new axioms. Then T will stay incomplete – unless the theory becomes ω-inconsistent or stops being nice.

Take the particular case where the sentence G, constructed as described in the last chapter, is added to PA as a new axiom. PA + G trivially entails G. But we can construct its own new Gödel sentence G′. And assuming PA + G is ω-consistent (which it surely is, as all its axioms are surely true on the standard interpretation built into L_A), neither G′ nor ¬G′ follows from our augmented theory. So neither can follow from the unaugmented original theory. In particular, G′ will be another true-but-unprovable sentence of PA, independent of the original Gödel sentence G. Repeating the argument gives us an unending stream of such independent unprovable sentences. And we can never 'fix up' PA by adding on enough new axioms to get a p.r. axiomatized, ω-consistent, negation-complete theory.

14.4 Rosser's improvement

So far, we've just been extracting results more or less directly to be mined from Gödel's great 1931 paper (at least when combined with the later observation that Q is p.r. adequate). In this section, however, we move on to note a clever construction due to Barkley Rosser (1936). He replaces the Gödel sentence G with a more complex cousin R; and then he shows that neither R nor ¬R is provable in PA, assuming just that PA is consistent. As with Gödel's construction the argument then generalizes. So Rosser improves Gödel's First Incompleteness Theorem, by enabling us to weaken the assumption of ω-consistency Gödel needed for half his Theorem and make do with plain consistency.

And frankly, that's all you really *need* to know at this point. However, for enthusiasts, we'd better tell more of the story (after all, the result is an important one).

So first, recall the familiar definition:

> $Gld(m, n)$ holds when m is the super g.n. for a PA proof of the diagonalization of the wff with g.n. n.

[6] For some details, see (Boolos et al., 2002, §§16.2, 16.4), comparing the theories *they* call **Q** and **R**. What's crucial for p.r. adequacy is that **Q** and **R** both deliver the Results that we listed in Section 7.2.

[7] If you worked through fn. 4 of the previous chapter, then you'll realize that we can immediately slightly sharpen Theorem 21/Theorem 21* with replacing talk of ω-consistency by talk of 1-consistency. But we won't fuss about this, given that Rosser's clever construction in the next section enables us to get rid of even the requirement of 1-consistency and make do with plain consistency.

And now let's next introduce a companion definition:

> $\overline{Gdl}(m, n)$ holds when m is the super g.n. for a PA proof of the *negation* of the diagonalization of the wff with g.n. n.

Both relations are p.r.; so both are capturable in PA, by a pair of wffs that we'll abbreviate by $\mathsf{Gdl}(\mathsf{x}, \mathsf{y})$ and $\overline{\mathsf{Gdl}}(\mathsf{x}, \mathsf{y})$ respectively.

Now consider the wff

$$\mathsf{Gdl}(\mathsf{x}, \mathsf{y}) \to (\exists \mathsf{v} \leq \mathsf{x})\overline{\mathsf{Gdl}}(\mathsf{v}, \mathsf{y})$$

Roughly, this wff says that if a number x codes a proof of a certain wff, then there is a smaller number v that codes a proof of the negation of the same wff.

And, from this point, we simply echo the Gödel construction. So, first step, we quantify to get

$$\forall \mathsf{x}(\mathsf{Gdl}(\mathsf{x}, \mathsf{y}) \to (\exists \mathsf{v} \leq \mathsf{x})\overline{\mathsf{Gdl}}(\mathsf{v}, \mathsf{y}))$$

Abbreviate this open wff as S (or $\mathsf{S}(\mathsf{y})$ when we want to emphasize the free variable), so its g.n. is $\ulcorner\mathsf{S}\urcorner$. Then, second step, diagonalize S to get

$$\mathsf{R} =_{\text{def}} \exists \mathsf{y}(\mathsf{y} = \ulcorner\mathsf{S}\urcorner \wedge \mathsf{S}(\mathsf{y}))$$

This is the Rosser sentence; it is trivially equivalent to

$$\forall \mathsf{x}(\mathsf{Gdl}(\mathsf{x}, \ulcorner\mathsf{S}\urcorner) \to (\exists \mathsf{v} \leq \mathsf{x})\overline{\mathsf{Gdl}}(\mathsf{v}, \ulcorner\mathsf{S}\urcorner))$$

Which reveals that the Rosser sentence is again of Goldbach type.[8]

Let's consider what R 'says' in the light of the fact that the formal predicates Gdl and $\overline{\mathsf{Gdl}}$ express the numerical relations *Gld* and $\overline{Gdl}$. Well, R is true when, for every number m, if $Gld(m, \ulcorner\mathsf{S}\urcorner)$ is true then there's a smaller number n such that $\overline{Gdl}(n, \ulcorner\mathsf{S}\urcorner)$. But by construction, $Gld(m, \ulcorner\mathsf{S}\urcorner)$ is true if m codes for a PA proof of R (think about it!). Similarly, $\overline{Gdl}(n, \ulcorner\mathsf{S}\urcorner)$ is true if n codes for a proof of $\neg\mathsf{R}$. Therefore R says that, for every number m, if m codes for a proof of R, there's a smaller number n that codes for a proof of $\neg\mathsf{R}$.

In sum, R says 'if R has a proof, there is also a "smaller" proof of $\neg\mathsf{R}$'.

In the light of this interpretation, R must be unprovable, assuming PA has true axioms and hence is consistent. For if R were provable, then it would be true. In other words, 'if R is provable, so is $\neg\mathsf{R}$' would be true; and this conditional would have a true antecedent, and so we can infer that $\neg\mathsf{R}$ is provable, which makes PA inconsistent, contrary to hypothesis. Therefore R is unprovable. Hence the material conditional 'if R is provable, so is $\neg\mathsf{R}$' has a false antecedent, so is true. In other words, R is true. Hence its negation $\neg\mathsf{R}$ is false, and therefore also unprovable (still assuming PA is sound). Hence R is another true-but-formally-undecidable wff.

[8] See Section 13.4. For note, R is equivalent to a quantification of an open wff which expresses a p.r. property – since both $\mathsf{Gdl}(\mathsf{x}, \ulcorner\mathsf{S}\urcorner)$ and $\overline{\mathsf{Gdl}}(\mathsf{x}, \ulcorner\mathsf{S}\urcorner)$ separately express p.r. properties, and applying the conditional truth-function and bounded quantification preserves primitive recursiveness (see Section 9.7).

However, to show that neither R nor $\neg\mathsf{R}$ is provable, we don't need the semantic assumption that PA is sound. And this time, as we said, we don't even need to assume that PA is ω-consistent. Assuming the mere consistency of PA is enough. So we have Rosser's strengthening of Theorem 19:

> **Theorem 22** *If* PA *is consistent, then there is an* L_A*-sentence* φ *of Goldbach type such that neither* φ *nor* $\neg\varphi$ *is provable in* PA.

The detailed proof of this theorem – often called the Gödel-Rosser Theorem – is a cousin of Gödel's proof in Section 13.5. However, it is a *lot* messier and less intuitive. I promise you won't miss anything of mathematical importance by entirely skipping the proof! But, for super-enthusiasts, here it is, in two parts.

Proof sketch: R *is unprovable if* PA *is consistent* Suppose R is provable. We'll prove a contradictory pair of propositions, (i) and (ii).

Given R is provable, some super g.n. m codes its proof. Hence, by definition, $Gld(m, \ulcorner\mathsf{S}\urcorner)$. Hence, since Gdl captures Gld, we have $\mathsf{PA} \vdash \mathsf{Gdl}(\overline{\mathsf{m}}, \ulcorner\mathsf{S}\urcorner)$. But by assumption, R is provable, so $\mathsf{PA} \vdash \forall \mathsf{x}(\mathsf{Gdl}(\mathsf{x}, \ulcorner\mathsf{S}\urcorner) \to (\exists \mathsf{v} \leq \mathsf{x})\overline{\mathsf{Gdl}}(\mathsf{v}, \ulcorner\mathsf{S}\urcorner))$.

Now instantiate the universal quantification by $\overline{\mathsf{m}}$, and then apply modus ponens to get $\mathsf{PA} \vdash (\exists \mathsf{v} \leq \overline{\mathsf{m}})\overline{\mathsf{Gdl}}(\mathsf{v}, \ulcorner\mathsf{S}\urcorner)$. By Result (3) about Q from Section 7.2, it follows easily that

(i) $\mathsf{PA} \vdash \overline{\mathsf{Gdl}}(0, \ulcorner\mathsf{S}\urcorner) \lor \overline{\mathsf{Gdl}}(\overline{1}, \ulcorner\mathsf{S}\urcorner) \lor \ldots \lor \overline{\mathsf{Gdl}}(\overline{\mathsf{m}}, \ulcorner\mathsf{S}\urcorner)$

However, given that PA is consistent, the same original assumption that R is provable implies that $\neg\mathsf{R}$ is not provable. That is to say, for every number n, $\overline{Gdl}(n, \ulcorner\mathsf{S}\urcorner)$. So, by the fact that $\overline{\mathsf{Gdl}}$ captures $\overline{Gdl}$, we have

(ii) for every n, $\mathsf{PA} \vdash \neg\overline{\mathsf{Gdl}}(\overline{\mathsf{n}}, \ulcorner\mathsf{S}\urcorner)$

But (ii) directly contradicts (i). So R is unprovable. ⊠

Proof sketch: $\neg\mathsf{R}$ *is unprovable if* PA *is consistent* Suppose $\neg\mathsf{R}$ is provable. Then there is some number n such that $\overline{Gdl}(n, \ulcorner\mathsf{S}\urcorner)$. Hence, $\mathsf{PA} \vdash \overline{\mathsf{Gdl}}(\overline{\mathsf{n}}, \ulcorner\mathsf{S}\urcorner)$. Whence, using an obvious shorthand, $\mathsf{PA} \vdash (\forall \mathsf{x} \geq \overline{\mathsf{n}})(\exists \mathsf{v} \leq \mathsf{x})\overline{\mathsf{Gdl}}(\mathsf{v}, \ulcorner\mathsf{S}\urcorner)$; and that trivially implies

(iii) $\mathsf{PA} \vdash (\forall \mathsf{x} \geq \overline{\mathsf{n}})(\mathsf{Gdl}(\mathsf{x}, \ulcorner\mathsf{S}\urcorner) \to (\exists \mathsf{v} \leq \mathsf{x})\overline{\mathsf{Gdl}}(\mathsf{v}, \ulcorner\mathsf{S}\urcorner))$

But if $\neg\mathsf{R}$ is provable, then (by consistency) R isn't. Hence for all numbers m, it isn't the case that $Gld(m, \ulcorner\mathsf{S}\urcorner)$. So, for all m, $\mathsf{PA} \vdash \neg\mathsf{Gdl}(\overline{\mathsf{m}}, \ulcorner\mathsf{S}\urcorner)$. Hence, trivially, for all m, $\mathsf{PA} \vdash \mathsf{Gdl}(\overline{\mathsf{m}}, \ulcorner\mathsf{S}\urcorner) \to (\exists \mathsf{v} \leq \overline{\mathsf{m}})\overline{\mathsf{Gdl}}(\mathsf{v}, \ulcorner\mathsf{S}\urcorner)$. Take the first $n + 1$ such results. Given these plus Result 4 about Q from Section 7.2, it follows that

(iv) $\mathsf{PA} \vdash (\forall \mathsf{x} \leq \overline{\mathsf{n}})(\mathsf{Gdl}(\mathsf{x}, \ulcorner\mathsf{S}\urcorner) \to (\exists \mathsf{v} \leq \mathsf{x})\overline{\mathsf{Gdl}}(\mathsf{v}, \ulcorner\mathsf{S}\urcorner))$

But we can show, using Result 8 from Section 7.2, that if we have both $\mathsf{PA} \vdash (\forall \mathsf{x} < \overline{\mathsf{n}})\varphi(\mathsf{x})$ and $\mathsf{PA} \vdash (\forall \mathsf{x} \geq \overline{\mathsf{n}})\varphi(\mathsf{x})$, then $\mathsf{PA} \vdash \forall \mathsf{x}\varphi(\mathsf{x})$. So from (iii) and (iv) it follows that

(v) $\mathsf{PA} \vdash \forall \mathsf{x}(\mathsf{Gdl}(\mathsf{x}, \ulcorner \mathsf{S} \urcorner) \to (\exists \mathsf{v} \leq \mathsf{x})\overline{\mathsf{Gdl}}(\mathsf{v}, \ulcorner \mathsf{S} \urcorner))$

But this last claim is just $\mathsf{PA} \vdash \mathsf{R}$, contradicting the joint assumptions that PA is consistent and $\neg\mathsf{R}$ is provable. So, keeping the consistency assumption, $\neg\mathsf{R}$ is unprovable. Phew! ☒

And, as with Gödel's original result, the result generalizes. Thus,

> **Theorem 23** *If T is a nice theory, then there is an arithmetic T-sentence φ of Goldbach type such that neither $T \vdash \varphi$ nor $T \vdash \neg\varphi$.*

So Rosser's clever trick nicely sharpens the First Incompleteness Theorem.

However, it doesn't really illuminate what is going on inside Gödel's proof. In the next chapter, then, we turn to digging deeper. But first, ...

14.5 Broadening the scope of the First Theorem

Our intuitive characterization of a properly formalized theory T says, essentially, that the property of being a T-proof must have a computable characteristic function (Sections 3.3 and 9.6). But we now know that not all computable functions are p.r. (Section 9.5). So it looks as if we could have an axiomatized theory (according to our intuitive characterization) which isn't p.r. axiomatized. Does this give us wriggle room to get around the First Theorem? Could there be a consistent formalized theory of arithmetic which was complete because not p.r. axiomatized?

True, a formalized theory T that isn't p.r. axiomatized will certainly be a rather peculiar beast. For checking that a putative T-proof *is* a proof will somehow have to involve a non-p.r. open-ended search, which will make it very unlike any familiar kind of axiomatized theory. Still, an oddly axiomatized theory which is complete would be better than no complete theory at all. But we can't achieve that either. For consider the following result which we'll prove in a moment:

> **Theorem 24** *If T is a formalized theory, then there is a p.r. axiomatized theory T' which has exactly the same theorems.*

This is an informal version of what's called Craig's Reaxiomatization Theorem: it says that if a body of wffs is axiomatizable, then is p.r. axiomatizable.[9]

Suppose, then, that T is a consistent formalized theory which includes Q. Our theorem tells us that there is a p.r. axiomatized theory T' which has the same theorems. But since it shares the same theorems, this p.r. axiomatized T' must also be consistent and include Q. So T' is nice, hence the Gödel-Rosser version of the First Theorem applies: there is an arithmetic sentence R such that $T' \nvdash \mathsf{R}$ and $T' \nvdash \neg\mathsf{R}$. Hence $T \nvdash \mathsf{R}$ and $T \nvdash \neg\mathsf{R}$.

[9] Jargon reminder: Σ is (p.r.) axiomatizable if there is a (p.r.) axiomatized formal theory T such that, for any wff φ, $\varphi \in \Sigma$ if and only if $T \vdash \varphi$. For the original theorem, see Craig (1953).

In short: *given Craig's theorem, Gödelian incompleteness infects* any *consistent formalized theory including enough arithmetic, whether it is p.r. axiomatized or not.* And here's why Craig's theorem holds:

Proof sketch If T is a formalized theory, its theorems can be effectively enumerated, by Theorem 2 (Section 3.5). So imagine stepping through some algorithmic procedure which effectively lists these theorems, and let's count the steps as we go along executing one minimal step at a time. Most of these steps are interim computations; just occasionally, the next theorem on the list will be printed out. Suppose that the theorem φ is produced at the step numbered $S(\varphi)$ of our algorithmic procedure.

Now for Craig's ingenious trick. Consider the derivative theory T' defined as follows: (i) for each T-theorem φ, T' has the axiom $(\varphi \wedge \varphi \wedge \varphi \wedge \ldots \wedge \varphi)$, with $S(\varphi)$ conjuncts; (ii) T''s sole rule of inference is $\wedge$-elimination.

Trivially, every T-theorem is a T'-theorem. And equally trivially, making the sole assumption that T has the usual rule of $\wedge$-introduction, every T'-theorem is a T-theorem.

Now note that given a wff ψ, we can tell whether it is a T'-axiom as follows. Stage 1: examine ψ and see if it is the n-fold conjunction of some particular wff φ (that can be done with 'for' loops). If it passes that test, count the n conjuncts and then move on to Stage 2: run the algorithm which lists theorems of T for steps 1 to n, and see if the algorithm then prints out that same φ (you can think of that as executing one big 'for' loop). If at the end of the process φ is indeed printed out, ψ is a T'-axiom, and otherwise it isn't.

But this means that testing whether ψ is a T'-axiom can be done just with 'for' loops. In other words, the chararacteristic function of the property of being a T'-axiom is primitive recursive. And therefore T' is p.r. axiomatized. ⊠

That's a rather quick-and-dirty proof. However we won't pause to tidy it up here. That's because later – when we have a general account of computation and decidability to hand – we'll be returning again to our more encompassing version of the First Theorem, and we will then be able to prove it more directly, without going via Craig's Theorem.

14.6 True Basic Arithmetic can't be axiomatized

Let's now note an immediate corollary of the first half of Theorem 21* combined with Craig's Theorem (in order to make good another claim we made in informal terms right back in Section 1.2, and to sharpen the result of Section 5.4).

Suppose L_A is, as before, the language of basic arithmetic. And let TA be the set of true sentences of L_A, i.e. closed wffs which are true on the standard interpretation built into L_A. So TA is True (Basic) Arithmetic. Then we have

> **Theorem 25** *There is no consistent axiomatized theory T whose set of theorems in the language L_A is exactly* TA.

Proof Suppose first that T is a p.r. axiomatized theory which entails all of TA. Then T will in particular prove the truths of basic arithmetic proved by Q which are enough to make Q p.r. adequate. So T is p.r. adequate. But then, if T is also consistent, it's nice: and we can use the Gödel construction to yield a sentence G in the language L_A which is true but unprovable-in-T, so there is a sentence in TA that T can't prove after all, contrary to hypothesis. So if T is consistent, it can't be p.r. axiomatized. And by Craig's Theorem, if it can't be p.r. axiomatized, then it can't be formally axiomatized at all. ☒

14.7 Incompletability – another quick look

Let's sum up. Suppose T is a nice theory. It's incomplete. Throw in some unprovable sentences as new axioms. As many as you want. Then, by the generalized Gödel-Rosser Theorem the resulting T^+ will still be incomplete, unless it stops being nice. But adding new axioms can't make a p.r. adequate theory any less adequate. So now we know the full price of T's becoming complete. Either (i) T ceases to be p.r. axiomatized or (ii) it becomes inconsistent. Outcome (i) is bad (for we now know that retreating to an axiomatized-but-not-p.r.-axiomatized theory won't get let us escape the First Theorem: T will have to stop being properly axiomatized at all). Outcome (ii) is worse.

Keep avoiding those bad outcomes, and T is incompletable.

15 Using the Diagonalization Lemma

When we are hill-walking, it often happens that we scramble up to the top of a peak, and then it becomes clear that we can descend a little (perhaps by a slightly different route) and from there easily strike out to reach some connected hill-tops.

We have now reached the peak of the First Incompleteness Theorem. Reviewing our route up, we can see that it takes us near a particular instance of the so-called 'Diagonalization Lemma'.[1] We will now backtrack to uncover this Lemma, and then note some of its important implications, which will take us up to a couple of neighbouring logical peaks. We start, however, by introducing ...

15.1 The provability predicate

Recall: $\mathit{Prfseq}(m, n)$ holds when m is the super g.n. for a PA-proof of the wff with g.n. n. So, introducing a natural further definition, $\mathit{Prov}(n) =_{\mathrm{def}} \exists v\, \mathit{Prfseq}(v, n)$ will hold when some number codes a proof of the wff with g.n. n – i.e. when n numbers a PA-theorem.

Since the relation *Prfseq* is p.r., it can be captured in PA by a wff $\mathsf{Prfseq}(\mathsf{x}, \mathsf{y})$. And now consider the wff $\mathsf{Prov}(\mathsf{x}) =_{\mathrm{def}} \exists \mathsf{v}\, \mathsf{Prfseq}(\mathsf{v}, \mathsf{x})$.[2] Evidently, this wff is true of n just when that n numbers a PA-theorem; hence $\mathsf{Prov}(\mathsf{x})$ *expresses* the provability property *Prov*.

Now generalize in the obvious way. For any theory T, here is similarly a relation $\mathit{Prfseq}_T(m, n)$ which holds when m is the super g.n. for a T-proof of the wff with g.n. n.[3] And if T is a nice theory (in the sense of Section 14.3, so it is p.r. axiomatized), then Prfseq_T will again be a p.r. relation. So there will be a corresponding wff $\mathsf{Prfseq}_T(\mathsf{x}, \mathsf{y})$ which captures this property in T. And we can now define $\mathsf{Prov}_T(\mathsf{x}) =_{\mathrm{def}} \exists \mathsf{v}\, \mathsf{Prfseq}_T(\mathsf{v}, \mathsf{x})$, where this complex *provability predicate* expresses the property of numbering a T-theorem.

Here are two simple facts about the provability predicate Prov_T. Assume that T is a nice theory and φ is some sentence; then

P1. if $T \vdash \varphi$, then $T \vdash \mathsf{Prov}_T(\ulcorner \varphi \urcorner)$;

P2. if T is ω-consistent, then if $T \nvdash \varphi$, then $T \nvdash \mathsf{Prov}_T(\ulcorner \varphi \urcorner)$.

[1] The name is conventional. Ordinarily, 'lemma' connotes a minor result: this Lemma is major theorem!

[2] We will assume, as we can without loss of generality, that $\mathsf{Prfseq}(\mathsf{x}, \mathsf{y})$ doesn't contain the variable 'v'.

[3] Of course, we are now talking in the context of some appropriate scheme for Gödel-numbering expressions of T – see the final remark of Section 12.1.

Proof of (P1) Suppose $T \vdash \varphi$. Then there is a T proof of the wff with g.n. $\ulcorner\varphi\urcorner$. Let this proof have the super g.n. m. Then, by definition, $\mathit{Prfseq}_T(m, \ulcorner\varphi\urcorner)$. Hence, since Prfseq_T is captured by Prfseq_T, it follows that $T \vdash \mathsf{Prfseq}_T(\overline{\mathsf{m}}, \ulcorner\varphi\urcorner)$. Hence $T \vdash \exists \mathsf{v}\, \mathsf{Prfseq}_T(\mathsf{v}, \ulcorner\varphi\urcorner)$, i.e. $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$. ⊠

Proof of (P2) Suppose $T \nvdash \varphi$. Then, for all m, not-$\mathit{Prfseq}_T(m, \ulcorner\varphi\urcorner)$. Since Prfseq_T is captured by Prfseq_T, it follows that for all m, $T \vdash \neg\mathsf{Prfseq}_T(\overline{\mathsf{m}}, \ulcorner\varphi\urcorner)$. So if $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$, i.e. $T \vdash \exists \mathsf{v}\, \mathsf{Prfseq}_T(\mathsf{v}, \ulcorner\varphi\urcorner)$, T would be ω-inconsistent. So contraposing, if T is ω-consistent, it can't prove $\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$. ⊠.

Note, by the way, that we need sharply to distinguish (P2) from

P3. if $T \nvdash \varphi$, then $T \vdash \neg\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$.

If (P1) and (P3) were both true, then Prov_T would not merely *express* but would also *capture* the provability property Prov_T. But as we'll soon see in Section 15.6, no wff can do that if T is nice. So (P3) is not always true. (A nice theory T is, so to speak, self-aware about what it *can* prove: but it doesn't know everything about what it *can't* prove.)

15.2 Diagonalization again

We saw that our Gödel sentence G for PA is constructed in such a way that it is true if and only if it is unprovable. Using the provability predicate Prov, we can now express this fact about G inside PA by the sentence

$$\mathsf{G} \leftrightarrow \neg\mathsf{Prov}(\ulcorner\mathsf{G}\urcorner)$$

Our next task is to show that we actually can *prove* this sentence in PA.

As a first step, let's think again about how our Gödel sentence for PA was constructed in Section 13.1. We'll do some manipulation to find an equivalent to G which explicitly involves Prov. So, recall,

$$\mathsf{G} =_{\mathrm{def}} \exists \mathsf{y}(\mathsf{y} = \ulcorner\mathsf{U}\urcorner \wedge \mathsf{U})$$

Here, '$\ulcorner\mathsf{U}\urcorner$' stands in for the numeral for U's g.n., and

$$\mathsf{U} =_{\mathrm{def}} \forall \mathsf{x} \neg\mathsf{Gdl}(\mathsf{x}, \mathsf{y})$$

where $\mathsf{Gdl}(\mathsf{x}, \mathsf{y})$ captures our old friend, the relation Gdl.

Now, as we saw in Section 12.6, we can put

$$\mathit{Gdl}(m, n) =_{\mathrm{def}} \mathit{Prfseq}(m, \mathit{diag}(n))$$

But the one-place p.r. function *diag* can be captured as a function in PA by some open wff $\mathsf{Diag}(\mathsf{x}, \mathsf{y})$. We can therefore give the following definition:

$$\mathsf{Gdl}(\mathsf{x}, \mathsf{y}) =_{\mathrm{def}} \exists \mathsf{z}(\mathsf{Prfseq}(\mathsf{x}, \mathsf{z}) \wedge \mathsf{Diag}(\mathsf{y}, \mathsf{z}))$$

And now to do some manipulation:

$$\begin{aligned}
\mathsf{U} &=_{\text{def}} \forall \mathsf{x}\neg\mathsf{Gdl}(\mathsf{x},\mathsf{y}) \\
&=_{\text{def}} \forall \mathsf{x}\neg\exists \mathsf{z}(\mathsf{Prfseq}(\mathsf{x},\mathsf{z}) \wedge \mathsf{Diag}(\mathsf{y},\mathsf{z})) && \text{(definition of Gdl)} \\
&\leftrightarrow \quad \forall \mathsf{x}\forall \mathsf{z}\neg(\mathsf{Prfseq}(\mathsf{x},\mathsf{z}) \wedge \mathsf{Diag}(\mathsf{y},\mathsf{z})) && \text{(pushing in the negation)} \\
&\leftrightarrow \quad \forall \mathsf{z}\forall \mathsf{x}\neg(\mathsf{Prfseq}(\mathsf{x},\mathsf{z}) \wedge \mathsf{Diag}(\mathsf{y},\mathsf{z})) && \text{(swapping quantifiers)} \\
&\leftrightarrow \quad \forall \mathsf{z}(\mathsf{Diag}(\mathsf{y},\mathsf{z}) \to \neg\exists \mathsf{x}\,\mathsf{Prfseq}(\mathsf{x},\mathsf{z})) && \text{(rearranging after `}\forall\mathsf{z}\text{')} \\
&\leftrightarrow \quad \forall \mathsf{z}(\mathsf{Diag}(\mathsf{y},\mathsf{z}) \to \neg\exists \mathsf{v}\,\mathsf{Prfseq}(\mathsf{v},\mathsf{z})) && \text{(changing variables)} \\
&=_{\text{def}} \forall \mathsf{z}(\mathsf{Diag}(\mathsf{y},\mathsf{z}) \to \neg\mathsf{Prov}(\mathsf{z})) && \text{(definition of Prov)} \\
&=_{\text{def}} \mathsf{U}' && \text{(new abbreviation)}
\end{aligned}$$

Two points about this. First, since U and U' are trivially equivalent, it isn't actually going to matter whether we work with G, the diagonalization of U, or G', the diagonalization of U'. These are distinct wffs involving different numerals: but it is an easy exercise to check that G' will do just as well as G for proving Gödel's theorem. We won't pause over this.

Second, to take us up to the theorem we want, note the U/U' equivalence is proved by simple logical manipulations which can be done inside the logic of PA. Which means in particular that

$$\mathsf{PA} \vdash \mathsf{U}(\ulcorner\mathsf{U}\urcorner) \leftrightarrow \mathsf{U}'(\ulcorner\mathsf{U}\urcorner)$$

But quite trivially,

$$\mathsf{PA} \vdash \mathsf{G} \leftrightarrow \mathsf{U}(\ulcorner\mathsf{U}\urcorner)$$

So putting things together, we have

$$\mathsf{PA} \vdash \mathsf{G} \leftrightarrow \mathsf{U}'(\ulcorner\mathsf{U}\urcorner)$$

Or unpacking this a bit,

$$\mathsf{PA} \vdash \mathsf{G} \leftrightarrow \forall\mathsf{z}(\mathsf{Diag}(\ulcorner\mathsf{U}\urcorner,\mathsf{z}) \to \neg\mathsf{Prov}(\mathsf{z}))$$

15.3 The Diagonalization Lemma: a special case

That last observation almost immediately gives us the desired result:

Theorem 26 $\mathsf{PA} \vdash \ \mathsf{G} \leftrightarrow \neg\mathsf{Prov}(\ulcorner\mathsf{G}\urcorner)$.

Proof Diagonalizing U yields G. Hence, by the definition of *diag*, we have $diag(\ulcorner\mathsf{U}\urcorner) = \ulcorner\mathsf{G}\urcorner$. Since by hypothesis Diag captures *diag*, it follows that

$$\mathsf{PA} \vdash \forall\mathsf{z}(\mathsf{Diag}(\ulcorner\mathsf{U}\urcorner,\mathsf{z}) \leftrightarrow \mathsf{z} = \ulcorner\mathsf{G}\urcorner)$$

But we've just noted that

$$\mathsf{PA} \vdash \mathsf{G} \leftrightarrow \forall\mathsf{z}(\mathsf{Diag}(\ulcorner\mathsf{U}\urcorner,\mathsf{z}) \to \neg\mathsf{Prov}(\mathsf{z}))$$

Hence, substituting our proven equivalent for $\mathsf{Diag}(\ulcorner\mathsf{U}\urcorner,\mathsf{z})$, it follows that

$$\mathsf{PA} \vdash \mathsf{G} \leftrightarrow \forall \mathsf{z}(\mathsf{z} = \ulcorner \mathsf{G} \urcorner \to \neg \mathsf{Prov}(\mathsf{z})).$$

But the right-hand side of that biconditional is trivially equivalent to $\neg\mathsf{Prov}(\ulcorner \mathsf{G} \urcorner)$. So we are done. ⊠

What this shows is that the informal claim 'G is true if and only if it is unprovable' can itself be formally proved within PA. Very neat!

And evidently the reasoning generalizes. If Prov_T is the provability predicate for T constructed analogously to the predicate Prov for PA, and if G_T is the analogously constructed Gödel sentence, then by exactly the same reasoning we have

> **Theorem 26*** *If T is a nice theory, $T \vdash \mathsf{G}_T \leftrightarrow \neg\mathsf{Prov}_T(\ulcorner \mathsf{G}_T \urcorner)$.*

15.4 The Diagonalization Lemma generalized

A little more reflection, however, shows that this last result is just a special case of a universal *Diagonalization Lemma*:[4]

> **Theorem 27** *If T is a nice theory and $\varphi(\mathsf{x})$ is* any *wff of its language with one free variable, then there is a sentence γ such that $T \vdash \gamma \leftrightarrow \varphi(\ulcorner \gamma \urcorner)$.*

Proof We use the same basic proof idea as before. In other words, we do to 'φ' pretty much what our Gödelian construction above did to '$\neg\mathsf{Prov}$'. So, first step, put $\psi(\mathsf{y}) =_{\mathrm{def}} \forall \mathsf{z}(\mathsf{Diag}_T(\mathsf{y}, \mathsf{z}) \to \varphi(\mathsf{z}))$.

Now let's do some diagonalization. So construct γ, the diagonalization of $\psi(\mathsf{y})$. This is equivalent to $\psi(\ulcorner \psi \urcorner)$. So $T \vdash \gamma \leftrightarrow \forall \mathsf{z}(\mathsf{Diag}_T(\ulcorner \psi \urcorner, \mathsf{z}) \to \varphi(\mathsf{z}))$.

Our theorem now follows speedily. Note that $\mathit{diag}_T(\ulcorner \psi \urcorner) = \ulcorner \gamma \urcorner$. Hence, $T \vdash \forall \mathsf{z}(\mathsf{Diag}_T(\ulcorner \psi \urcorner, \mathsf{z}) \leftrightarrow \mathsf{z} = \ulcorner \gamma \urcorner)$, since by hypothesis Diag_T captures diag_T.

Hence $T \vdash \forall \mathsf{z}(\mathsf{Diag}_T(\ulcorner \psi \urcorner, \mathsf{z}) \to \varphi(\mathsf{z})) \leftrightarrow \forall \mathsf{z}(\mathsf{z} = \ulcorner \gamma \urcorner \to \varphi(\mathsf{z}))$, since we've just proved that the antecedents of the conditionals on either side are equivalents. But the left-hand side of our new biconditional is equivalent to γ, and the right-hand side is in turn equivalent to $\varphi(\ulcorner \gamma \urcorner)$. So $T \vdash \gamma \leftrightarrow \varphi(\ulcorner \gamma \urcorner)$. ⊠

We will use this really beautiful Lemma three times over in the rest of this chapter, first in revisiting the incompleteness result, and then to prove two further results. Later in the book, we'll return to the Lemma again.

But first a quick remark about jargon. Suppose that the function f maps the argument a back to a itself, so that $f(a) = a$: then a is said to be a *fixed point* for f. And a theorem to the effect that, under certain conditions, there is a fixed point for f is a *fixed-point theorem*. By analogy, Theorem 27 is often also referred to as a fixed point theorem, with γ as a 'fixed point' for the predicate $\varphi(\mathsf{z})$.

[4] In a footnote added to later reprintings of his (1934), Gödel says that this Lemma 'was first noted by Carnap (1934)': first noted in print, yes; but Gödel himself had probably already got there in 1930.

15.5 Incompleteness again

Now note the following result, which we prove using the principles (P1) and (P2) we established at the beginning of the chapter:

> **Theorem 28** *Let T be a nice theory, and let γ be* any *fixed point for $\neg\mathsf{Prov}_T(\mathsf{z})$. Then, if T is consistent, $T \nvdash \gamma$, and if T is ω-consistent, then $T \nvdash \neg\gamma$.*

Proof For readability, let's start dropping subscript 'T's when context readily supplies them. Now, γ is a fixed point for $\neg\mathsf{Prov}$, i.e. $T \vdash \gamma \leftrightarrow \neg\mathsf{Prov}(\ulcorner\gamma\urcorner)$. So if $T \vdash \gamma$ then $T \vdash \neg\mathsf{Prov}(\ulcorner\gamma\urcorner)$. But, by (P1), we also have if $T \vdash \gamma$ then $T \vdash \mathsf{Prov}(\ulcorner\gamma\urcorner)$. So, if T is consistent, we can't have $T \vdash \gamma$.

Now assume T is ω-consistent (and hence consistent), and suppose $T \vdash \neg\gamma$. Since $T \vdash \gamma \leftrightarrow \neg\mathsf{Prov}(\ulcorner\gamma\urcorner)$, it follows $T \vdash \mathsf{Prov}(\ulcorner\gamma\urcorner)$. But by consistency, if $T \vdash \neg\gamma$, then $T \nvdash \gamma$. Hence by (P2), $T \nvdash \mathsf{Prov}(\ulcorner\gamma\urcorner)$. Contradiction. So, assuming T *is* ω-consistent, we can't have $T \vdash \neg\gamma$. ⊠

In sum, Theorem 28 tells us that if there *is* a fixed point for $\neg\mathsf{Prov}$, then T can't be negation-complete, assuming it is nice and ω-consistent. But Theorem 27 tells us that such a fixed point must indeed exist (and Theorem 26* tells us where to find it). Put the results together and we've got to incompleteness again – which gives us nice reworking of the themes in Gödel's proof.[5]

15.6 Capturing provability?

Having backtracked to the Diagonalization Lemma, and re-established incompleteness using that, we can now strike out again to get a range of new theorems. We'll mention just two in the rest of this chapter.

First, consider again the T-wff $\mathsf{Prov}_T(\mathsf{x})$ which *expresses* the property $Prov_T$ of being the g.n. of a T-theorem. The obvious next question to ask is: does this wff also case-by-case *capture* that property? The following theorem shows that it can't:

> **Theorem 29** *No open wff in a nice theory T can capture the corresponding numerical property* Prov_T.

Proof Suppose for reductio that $\mathsf{P}(\mathsf{x})$ abbreviates an open wff – not necessarily identical to $\mathsf{Prov}_T(\mathsf{x})$ – which captures $Prov_T$. By the Diagonalization Lemma applied to $\neg\mathsf{P}(\mathsf{z})$, there is some wff γ such that

[5] And we can fairly easily extract from the proof of the Diagonalization Lemma the further point that there's an unprovable fixed point of Goldbach type.

We can then go on similarly to get the Gödel-Rosser theorem by applying the Diagonalization Lemma again, this time to a more complex wff, and then showing that a fixed point for this more complex wff is unprovable in a consistent theory. But this doesn't reveal anything new, so this time we'll leave the details as an exercise for enthusiasts to work out.

(i) $T \vdash \gamma \leftrightarrow \neg\mathsf{P}(\ulcorner\gamma\urcorner)$.

By the general assumption that P captures $Prov_T$, we have in particular

(ii) if $T \vdash \gamma$, i.e. $Prov_T(\ulcorner\gamma\urcorner)$, then $T \vdash \mathsf{P}(\ulcorner\gamma\urcorner)$
(iii) if $T \nvdash \gamma$, i.e. not-$Prov_T(\ulcorner\gamma\urcorner)$, then $T \vdash \neg\mathsf{P}(\ulcorner\gamma\urcorner)$

Contradiction quickly follows. By (iii) and (i), if $T \nvdash \gamma$, then $T \vdash \gamma$. Hence $T \vdash \gamma$. So by (ii) and (i) we have both $T \vdash \mathsf{P}(\ulcorner\gamma\urcorner)$ and $T \vdash \neg\mathsf{P}(\ulcorner\gamma\urcorner)$ making T inconsistent, contrary to hypothesis. ⊠

Hence $Prov_T$ cannot be captured in T: and so – to answer our original question – $\mathsf{Prov(x)}_T$ in particular doesn't capture that property.

But now recall that T, being p.r. adequate, can capture any p.r. property. So it immediately follows that the provability property $Prov_T$ for any nice theory is not p.r.: which is to say that there is no p.r. function of n which returns 0 if $Prov_T(n)$ is true, and 1 otherwise. In other words, there is no p.r. function which decides what is (the code number of) a theorem of T.

Later, we will be able to improve this result and prove an important theorem: there's *no* mechanical way at all of deciding whether a wff is a theorem of a nice theory T, whether by a p.r. computation or by any other kind of algorithmic procedure. But we aren't yet in a position to go on to give a formal proof of this stronger claim, because we haven't yet got to hand a general theory of algorithmic procedures.[6] We'll see how to get the stronger result in Section **??**.

15.7 Tarski's Theorem

Finally, we visit twin peaks which can also be reached very quickly via the Diagonalization Lemma. The path is very straightforward, but it leads to a pair of spectacular results that are usually packaged together as *Tarski's Theorem*.[7]

(a) Recall: 'snow is white' is true iff snow *is* white. Likewise for all other sensible replacements for 'snow is white'. And that's because of the meaning of 'true'.

This observation motivates the following definition. Suppose that L is any language that includes a bit of arithmetic, and so within L we can use the numeral '$\ulcorner\varphi\urcorner$' to pick out the sentence φ via Gödel coding. Then we'll say the open L-wff $\mathsf{T(x)}$ is a *truth-predicate* just in case every instance of

1. $\mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$

is true.

Next, we'll say that a theory T includes a *truth-theory* for L if for some $\mathsf{T(x)}$,

[6] Compare our informal Theorem 4 back in Section 5.2, which shows that, for any consistent sufficiently strong T, *Prov* can't be an algorithmically decidable property of numbers.

[7] Alfred Tarski investigated these matters in his classic (1933); though Gödel again had already noted the key point, e.g. in a letter to Zermelo written in October, 1931 (Gödel, 2003, pp. 423–429). Also see the quotation later in this section.

2. for every L-wff φ, $T \vdash \mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$

Equally often, a truth-theory for L is called a 'definition of truth for L'.

Suppose then that T is a nice arithmetical theory. An obvious question arises: could T be competent to define truth for its own language? And the answer is immediate:

> **Theorem 30** *No nice theory can define truth for its own language.*

Proof Suppose there is a predicate $\mathsf{T}(\mathsf{x})$ such that (2) holds. Since T is nice, the Diagonalization Lemma applies, so applying the Lemma to the negation of $\mathsf{T}(\mathsf{x})$, we know that there must be some L_A sentence L – a Liar sentence! – such that

3. $T \vdash \mathsf{L} \leftrightarrow \neg\mathsf{T}(\ulcorner\mathsf{L}\urcorner)$

But then we also have the following instance of (2):

4. $T \vdash \mathsf{T}(\ulcorner\mathsf{L}\urcorner) \leftrightarrow \mathsf{L}$

It is immediate that T is inconsistent and so not nice, contrary to hypothesis. So there can't be a predicate such that (2) holds after all. ⊠

(b) That first theorem puts limits on what a nice theory can *prove* about truth. But with very modest extra assumptions, we can put limits on what a theory's language can even *express* about truth.

Consider L_A for the moment. Define the numerical property *True* as follows: *True*(n) if and only if n is the g.n. of a true L_A-sentence. And suppose, for reductio, that the open L_A-wff $\mathsf{T}(\mathsf{x})$ expresses *True*. Then, by definition, for all L_A-sentences φ, $\mathsf{T}(\ulcorner\varphi\urcorner)$ holds iff *True*$(\ulcorner\varphi\urcorner)$, i.e. iff φ is true. In other words, each instance of a formal wff of the shape

1. $\mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$

will be a truth (exactly as you'd expect, then, expressing *True* makes a predicate a truth-predicate in the sense we defined).

But now remember that the Diagonalization Lemma holds for any nice theory T. Hence it holds for Q in particular. So, applying the Lemma to the negation of $\mathsf{T}(\mathsf{x})$, we know that for some L_A sentence L, $\mathsf{Q} \vdash \mathsf{L} \leftrightarrow \neg\mathsf{T}(\ulcorner\mathsf{L}\urcorner)$. But (and here comes the extra assumption we said we were going to invoke!) everything Q proves is true, since Q's axioms are of course true. So

2. $\mathsf{L} \leftrightarrow \neg\mathsf{T}(\ulcorner\mathsf{L}\urcorner)$

will also be a true L_A wff. However, as a special case of (1),

3. $\mathsf{T}(\ulcorner\mathsf{L}\urcorner) \leftrightarrow \mathsf{L}$

must be true too. But (2) and (3) immediately lead to $\mathsf{L} \leftrightarrow \neg\mathsf{L}$ and hence contradiction. Therefore our supposition that (1) is always true has to be rejected. Hence no predicate of L_A can even express the numerical property *True*.

The argument evidently generalizes. Take any language L rich enough for us to be able to formulate in L something equivalent to the very elementary arithmetical theory Q (that's so we can prove the Diagonal Lemma again). Call that an adequate arithmetical language. Then by the same argument, assuming Q is a correct theory,

> **Theorem 31** *No predicate of an adequate arithmetical language L can express the numerical property* True_L *(i.e. the property of numbering a truth of L).*

This tells us that while you can express *syntactic* properties of a sufficiently rich formal theory of arithmetic (like provability) inside the theory itself via Gödel numbering, you can't express some key *semantic* properties (like arithmetical truth) inside the theory.

And this last observation of course gives us yet another, particularly illuminating, argument for incompleteness. For example, truth in L_A isn't provability in PA: for while PA-provability is expressible, truth isn't. So assuming that PA is sound and everything provable in it is true, this means that there must be truths of L_A which it can't prove. Similarly for other p.r. axiomatized theories.

We might well take this to be *the* master argument for incompleteness.[8] Gödel himself wrote (in response to a query)

> I think the theorem of mine that von Neumann refers to is . . . that a complete epistemological description of a language A cannot be given in the same language A, because the concept of truth of sentences in A cannot be defined in A. It is this theorem which is the true reason for the existence of undecidable propositions in the formal systems containing arithmetic. I did not, however, formulate it explicitly in my paper of 1931 but only in my Princeton lectures of 1934. The same theorem was proved by Tarski in his paper on the concept of truth [Tarski (1933)].[9]

(c) We'll return to issues about truth later. For the moment, just note that we have the following situation. Suppose T is a nice theory, then (1) there are some numerical properties that T can capture (the p.r. ones for a start); (2) there are some properties that T can express but not capture (for example Prov_T); and (3) there are some properties that T's language cannot even express (for example True_T, the numerical property of numbering-a-true-T-wff).

[8] Or if not *the* argument, at least one of the proofs which, so to speak, belongs in The Book in which God maintains the best proofs for mathematical theorems. For Paul Erdős's conceit of proofs from The Book, see Aigner and Zielger (2004).

[9] Gödel's letter is quoted by Feferman in his (1984), which also has a very interesting discussion of why Gödel chose not to highlight this line of argument for incompleteness in his original paper.

It is not, we should hasten to add, that the property True_T is mysteriously ineffable, and escapes all formal treatment. A richer theory T^+ with a richer language may perfectly well be able to capture True_T. But the point remains that, however rich a given theory of arithmetic is, there will be limitations, not only on what numerical properties it can capture but even on which numerical properties that particular theory's language can express.

Interlude: about the First Theorem

We have achieved our first main goal, namely to prove Gödel's First Incompleteness Theorem. It will do no harm to pause for breath and quickly survey what we've established and how we established it. Equally importantly, we should make it clear what we *haven't* proved. The Theorem seems inevitably to attract serious misunderstandings. Let's also briefly try to block just a few of these.[1]

(a) To begin with what we *have* shown (in headline terms). Suppose we are trying to regiment the truths of basic arithmetic – i.e. the truths expressible in terms of successor, addition, multiplication, and the apparatus of first-order logic. Ideally, we'd like to construct a consistent theory T which entails *all* these truths: so we'd like T to be negation-complete. But, given some entirely natural assumptions about the sort of theory we want, we can prove that this ideal cannot be achieved.

The first natural assumption is that T should be set up so that it is effectively decidable whether a putative T-proof really *is* a proof. Indeed, we surely want more: we want it to be straightforwardly decidable what's a proof, without needing open-ended search procedures. Sharpened up, this is the assumption that T should be a p.r. axiomatized theory.

But next, there's a fork in the path. Do we assume T is a sound theory (with true axioms and a truth-preserving deductive system) or not?

If we assume soundness – and what more natural than the assumption that our theory which aims to generate all truths of basic arithmetic should itself be truthful? – then we have a pretty easy *semantic* argument for incompleteness:

1. Every p.r. function and relation can be *expressed* in L_A, the language of basic arithmetic. (Theorem 14, proved in Sections 11.2 to 11.4 – the argument is elementary once we grasp the β-function trick.)

2. The relation $Gld_T(m, n)$, which holds when m codes for a T-proof of the diagonalization of the wff with code number n, is primitive recursive – on the assumptions that T is p.r. axiomatized and that we have a sensible coding system. (Gld_T is definable in terms of $Prfseq_T$ and a trivially p.r. $diag_T$ function: so we can just appeal to Theorem 17, for which we gave a quick and dirty but entirely accessible proof in Section 12.3.)

3. Then, using the fact that Gdl_T must be expressible in L_A, we can construct a wff G_T which is true if and only if it is unprovable. (Section 13.1:

[1] We haven't space to explode all the wilder misconceptions about the Theorem. See Franzén (2005) for a more wide-ranging demolition job.

the construction is ingenious, but quite easy to understand.)

4. Given Gdl_T is true if and only if it is unprovable, there is then an entirely straightforward argument to the conclusion that, if T is sound, then $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$. (Use the simple argument which we first gave, in fact, in Section 1.2.)

If on the other hand, we don't assume that T is sound, and want to make do with much weaker non-semantic assumptions about consistency, then we make life quite a lot harder for ourselves. We can put one version of the *syntactic* argument as follows (this way of putting things highlights the parallels and the contrasts with our semantic argument):

$1'$. Assume T contains at least as much arithmetic as Q. Then every p.r. function and relation can be *captured* in T. (Theorem 16, which rests on two other results. (i) We sharpen the theorem that every p.r. function is expressible in L_A to the theorem that that every p.r. function is in fact expressible using a Σ_1 wff of L_A. And (ii) we show that Q can capture any function expressible by a Σ_1 wff.)

$2'$. As before, $Gld_T(m, n)$ is primitive recursive.

$3'$. Then, using the fact that T can *capture* Gdl_T, we then construct a wff G_T for which we can then show that T *proves* the formal wff which states: G_T if and only if it is unprovable. (Theorem 26, a special case of the Diagonalization Lemma.)

$4'$. Given T can prove that Gdl_T is true if and only if it is unprovable, we can then show: if T is consistent, $T \nvdash \mathsf{G}_T$; and if T is ω-consistent, $T \nvdash \neg\mathsf{G}_T$. (Section 15.5.)

Note that each of stages $1'$, $2'$ and $4'$ of the semantic argument is significantly harder than its counterpart in the first syntactic argument. That's a real cost. But we get a real benefit: we get a stronger incompleteness result, one that depends only on assumptions about T's consistency rather than an assumption about soundness. It is some version of this stronger result which is normally called Gödel's First Theorem. And at the cost of additional complications, we can use the Rosser variant on our syntactic argument to weaken the assumption of ω-consistency to that of mere consistency, to get the Gödel-Rosser Theorem: to use our summary shorthand, if a theory is nice, it can't be complete.

Still, given that we surely *do* want to be working with *sound* theories, you might reasonably ask: is the possibility of an argument resting on weaker consistency assumptions of more than technical, mathematical, interest?

Yes. (i) As we'll explain in the next Interlude, one of Gödel's aims was to confront Hilbert's Programme, which gives us a context where we need to focus on consistency rather than soundness assumptions. And (ii) as we'll see before that, in the very next chapter, Gödel's *Second* Incompleteness Theorem emerges from reflecting on our *syntactic* argument.

(b) It is often said that these now familiar proofs of incompleteness depend on *self-reference*. But we must be careful here. As we stressed before, the Gödel sentence G_T – which has the form $\exists \mathsf{y}(\mathsf{y} = \ulcorner \mathsf{U} \urcorner \wedge \mathsf{U}(\mathsf{y}))$ – is semantically speaking an entirely normal sentence of L_A. In particular, the embedded numeral $\ulcorner \mathsf{U} \urcorner$ refers to a number, not to some sentence. So G_T isn't self-referential in a strict sense (see Section 13.2).

Still, the long and complicated predicate U is of course constructed with an eye on some Gödel numbering convention. Then, when we diagonalize, we use the numeral $\ulcorner \mathsf{U} \urcorner$ which denotes the number which is fact the code number for U on the *same* Gödel numbering convention. Call this, if you like, 'self-reference-via-coding'. And of course, it is the background Gödel numbering convention which guides our construction which gives the resulting G_T its importance. Purely arithmetically, G_T is just a horribly long sentence of no evident interest.

These quick and obvious remarks suggest two related questions:

1. Can we find unprovable sentences of, e.g., PA (our benchmark first-order theory of arithmetic) which *are* of purely arithmetical interest? Or at least, can we find unprovable sentences of PA which are of some intrinsic mathematical interest? Or are the unprovable sentences all rather like G, sentences whose interest depends on coding tricks brought in from outside arithmetic?

2. Do proofs of incompleteness always depend on self-reference-via-coding or some close analogue?

Both these questions are rather vaguely posed (what counts as having 'purely arithmetical interest'? what counts as a 'close analogue' of an argument depending on self-reference-via-coding?). However, we'll see later that there *is* a variety of significantly different proofs of incompleteness. Though we will also argue that while some of the various sentences that cannot be decided in PA may be of intrinsic mathematical interest, perhaps none of them are of purely arithmetical interest. We'll return to these matters.

(c) What do our core technical results show? Let's run through a few supposed implications:

(i) *'There are arithmetical truths which are not formally provable in* any *nice theory'.* Not so. Take any arithmetical truth φ; this is, for a start, trivially provable in the formal theory $\mathsf{Q} + \varphi$ (i.e. the theory we get by adding φ as a new axiom to Q). But that theory is still nice. It is consistent (assuming Q is sound, which it is); it is p.r. axiomatized (since has still got a finite list of axioms); it is p.r. adequate (because Q is).

(ii) *'For any nice theory T, there will be truths that T can't prove but we can.'* Not so. Suppose G_T is a Gödel-sentence for T. Then $T \nvdash \mathsf{G}_T$. If T is consistent,

G_T will be true. And we can prove that fact, i.e. prove that *if* T is consistent, G_T is true. But this doesn't yield a proof of G_T, unless we have a proof that T is consistent. And very often, we won't have (e.g. because T is too complicated).[2]

(iii) *'For some nice theories T, there are truths that T can't prove but we can: and these truths can't be proved in a formal theory we have reason to accept.* This is more cautious. First, it acknowledges that we can only establish *some* Gödel sentences to be true; second it acknowledges that given any truth we can concoct *some* formal theory that proves it (hence the restriction to 'natural' theories which we have a reason to accept). But it's still wrong.

For take the reasoning that we run through, from outside the theory T, to convince ourselves that G_T is true, when we can do this. Then there's nothing in the First Theorem to stop us locating the principles involved in this reasoning, and so constructing a formal theory (distinct from T) which encapsulates these principles, in which we prove G_T to be true. But we have good reason to accept *that* formal theory, since it just regiments some aspects of how we reason anyway.

(d) So what *can* we infer from the First Theorem that if a theory is nice, it can't be complete? Well, that simple but deep Gödelian fact does sabotage, once and for all, any project of trying to show that all basic arithmetical truths can be thought of as deducible, in some standard deductive system, from some one set of fundamental axioms which can be specified in a tidy, p.r., way.

And it therefore sabotages the logicist project in particular, at least in its classic form of aiming to deduce all arithmetic from a tidy set of principles which are either logical axioms or definitions. Which at first sight seems a major blow; for there surely *is* something very deeply attractive about the thought that in exploring the truths of basic arithmetic, we are just exploring the deductive consequences of the axiomatic truths we grasp in grasping the very ideas of the natural number series and the of fundamental operations of addition and multiplication. Later, we'll encounter a couple of fall-back positions for the logicist which aim, in different ways, to preserve some of this core thought.

For the moment, then, let's finish with the following observation. If the truths of even basic arithmetic run beyond what is provable in any given formal system which regiments a set of p.r. specifiable axioms and a set of deductive procedures, then even arithmetic is – so to speak – inexhaustible. Mathematicians are not going to run out of work, as they develop ever richer formal settings in which to prove more truths. Which is, perhaps, a rather hopeful thought, and which introduces another theme we will need to return to later.

But first, however, we need to move on to get more technicalities under our belt. It is time to explore the Second Theorem.

[2] And the threat of inconsistency is not, in the general case, merely notional. To mention a famous example, towards the end of the first edition of his *Mathematical Logic* (1940), W. V. Quine proves Gödel's theorem for his system. Should we then conclude that the relevant Gödel sentence is true? We go wrong if we do: for it turns out that Quine's formal theory in that edition is inconsistent!

Bibliography

Gödel's papers are identified by their dates of first publication; but translated titles are used and references are to the versions in the *Collected Works*, where details of the original publications are to be found. Similarly, articles or books by Frege, Hilbert etc. are identified by their original dates, but references are whenever possible to standard English translations.

Aigner, M. and Zielger, G. M., 2004. *Proofs from The Book.* Berlin: Springer, 3rd edn.

Balaguer, M., 1998. *Platonism and Anti-Platonism in Mathematics.* New York: Oxford University Press.

Boolos, G., Burgess, J., and Jeffrey, R., 2002. *Computability and Logic.* Cambridge: Cambridge University Press, 4th edn.

Cantor, G., 1874. On a property of the set of real algebraic numbers. In Ewald 1996, Vol. 2, pp. 839–843.

Cantor, G., 1891. On an elementary question in the theory of manifolds. In Ewald 1996, Vol. 2, pp. 920–922.

Carnap, R., 1934. *Logische Syntax der Sprache.* Vienna: Springer. Translated into English as Carnap 1937.

Carnap, R., 1937. *The Logical Syntax of Language.* London: Paul, Trench.

Carnap, R., 1950. *Logical Foundations of Probability.* Chicago: Chicago University Press.

Chabert, J.-L. (ed.), 1999. *A History of Algorithms.* Berlin: Springer.

Church, A., 1936. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58: 345–363.

Church, A., 1937. Review of Turing 1936. *Journal of Symbolic Logic*, 2: 42–43.

Cooper, S. B., 2004. *Computability Theory.* Boca Raton, Florida: Chapman and Hall/CRC.

Craig, W., 1953. On axiomatizability within a system. *Journal of Symbolic Logic*, 18: 30–32.

Cutland, N. J., 1980. *Computability.* Cambridge: Cambridge University Press.

Davis, M., 1982. Why Gödel didn't have Church's Thesis. *Information and Control*, 54: 3–24.

Dedekind, R., 1888. Was sind und was sollen die Zallen? In Ewald 1996, Vol. 2, pp. 790–833.

Earman, J., 1995. *Bangs, Crunches, Whimpers, and Shrieks: Singularities and Acausalities in Relativistic Spacetimes.* New York: Oxford University Press.

Enderton, H. B. (ed.), 2002. *A Mathematical Introduction to Logic.* San Diego: Academic Press.

Epstein, R. L. and Carnielli, W. A., 2000. *Computability: Computable Functions, Logic, and the Foundations of Mathematics.* Wadsworth.

Ewald, W. (ed.), 1996. *From Kant to Hilbert.* Oxford: Clarendon Press.

Feferman, S., 1984. Kurt gödel: conviction and caution. *Philosophia Naturalis*, 21: 546–562. In Feferman 1998, pp. 150–164.

Feferman, S., 1998. *In the Light of Reason.* New York: Oxford University Press.

Field, H., 1989. *Realism, Mathematics and Modality.* Oxford: Basil Blackwell.

Fisher, A., 1982. *Formal Number Theory and Computability.* Oxford: Clarendon Press.

Franzén, T., 2005. *Gödel's Theorem: An Incomplete Guide to its Use and Abuse.* Wellesley, MA: A. K. Peters.

Frege, G., 1882. On the scientific justification of a conceptual notation. In Frege 1972, pp. 83–89.

Frege, G., 1884. *Die Grundlagen der Arithmetik.* Breslau: Verlag von Wilhelm Koebner. Translated as Frege 1950.

Frege, G., 1891. Function and concept. In Frege 1984, pp. 137–156.

Frege, G., 1950. *The Foundations of Arithmetic.* Basil Blackwell.

Frege, G., 1964. *The Basic Laws of Arithmetic.* Berkeley and Los Angeles: University of California Press.

Frege, G., 1972. *Conceptual Notation and related articles.* Oxford: Clarendon Press. Edited by Terrell Ward Bynum.

Gandy, R., 1988. The confluence of ideas in 1936. In R. Herken (ed.), *The Universal Turing Machine*, pp. 55–111. Oxford University Press.

Gödel, K., 1929. On the completeness of the calculus of logic. In Gödel 1986, Vol. 1, pp. 60–101.

Gödel, K., 1931. On formally undecidable propositions of *Principia Mathematica* and related systems I. In Gödel 1986, Vol. 1, pp. 144–195.

Gödel, K., 1934. On undecidable propositions of formal mathematical systems. In Gödel 1986, Vol. 1, pp. 346–371.

Gödel, K., 1986. *Collected Works, Vol. 1: Publications 1929–1936.* New York and Oxford: Oxford University Press.

Gödel, K., 2003. *Collected Works, Vol. 5: Correspondence H–Z.* Oxford: Clarendon Press.

Hájek, P. and Pudlák, P., 1993. *Metamathematics of First-Order Arithmetic.* Berlin: Springer.

Hilbert, D., 1918. Axiomatic thought. In Ewald 1996, Vol. 2, pp. 1107–1115.

Hilbert, D. and Ackermann, W., 1928. *Grundzüge der Theoretischen Logik.* Berlin: Springer. 2nd edn. 1938, translated as *Principles of Mathematical Logic*, New York: Chesea Publishing Co., 1950.

Hilbert, D. and Bernays, P., 1934. *Grundlagen der Mathematik, Vol I.* Berlin: Springer.

Hunter, G., 1971. *Metalogic.* London: Macmillan.

Isles, D., 1992. What evidence is there that 2^{65536} is a natural number? *Notre Dame Journal of Formal Logic*, 33: 465–480.

Kleene, S. C., 1936a. General recursive functions of natural numbers. *Mathematische Annalen*, 112: 727–742.

Kleene, S. C., 1936b. λ-definability and recursiveness. *Duke Mathematical Journal*, 2: 340–353.

Kleene, S. C., 1952. *Introduction to Metamathematics*. Amsterdam: North-Holland Publishing Co.

Kleene, S. C., 1967. *Mathematical Logic*. New York: John Wiley.

Kleene, S. C., 1981. Origins of recursive function theory. *Annals of the History of Computing*, 3: 52–67.

Lakatos, I., 1976. *Proofs and Refutations*. Cambridge: Cambridge University Press.

Leary, C. C., 2000. *A Friendly Introduction to Mathematical Logic*. New Jersey: Prentice Hall.

Lindström, P., 2003. *Aspects of Incompleteness*. A. K. Peters, 2nd edn.

Mendelson, E., 1997. *Introduction to Mathematical Logic*. Chapman and Hall, 4th edn.

Meyer, A. R. and Ritchie, D., 1967. Computational complexity and program structure. Tech. Rep. RC-1817, IBM.

Peano, G., 1889. *The Principles of Arithmetic*. In van Heijenoort 1967, pp. 85–97.

Péter, R., 1934. Über den zusammenhang der verschiedenen Begriffe der rekursiven Funktionen. *Mathematische Annalen*, 110: 612–632.

Potter, M., 2004. *Set Theory and its Philosophy*. Oxford: Oxford University Press.

Quine, W. V., 1940. *Mathematical Logic*. Cambridge, MA: Harvard University Press.

Robinson, R., 1952. An essentially undecidable axiom system. In *Proceedings of the International Congress of Mathematicians, Cambridge, Mass., 1950, Vol. 1*, pp. 729–730. Providence, R.I.

Rosser, J. B., 1936. Extensions of some theorems of Gödel and Church. *Journal of Symbolic Logic*, pp. 230–235.

Russell, B., 1902. Letter to Frege. In van Heijenoort 1967, pp. 124–125.

Russell, B., 1903. *The Principles of Mathematics*. London: George Allen and Unwin.

Russell, B. and Whitehead, A. N., 1910–13. *Principia Mathematica*. Cambridge: Cambridge University Press.

Shepherdson, J. C. and Sturgis, H. C., 1963. Computability of recursive functions. *Journal of the Association for Computing Machinery*, 10: 217–255.

Sieg, W., 1997. Step by recursive step: Church's analysis of effective calculability. *Bulletin of Symbolic Logic*, 3: 154–180.

Skolem, T., 1923. The foundations of elementary arithmetic established by means of the recursive mode of thought, without the use of apparent variables ranging over infinite domains. In van Heijenoort 1967, pp. 303–333.

Smorynski, C., 1977. The incompleteness theorems. In J. Barwise (ed.), *Handbook of Mathematical Logic*, pp. 821–865. Amsterdam: North-Holland.

Smullyan, R. M., 1992. *Gödel's Incompleteness Theorems*. Oxford: Oxford University Press.

Tarski, A., 1933. *Pojecie Prawdy w Jezykach Nauk Dedukcyjnych*. Warsaw. Translated into English in Tarksi 1956, pp. 152–278.

Bibliography

Tarski, A., 1956. *Logic, Semantics, Metamathematics.* Oxford: Clarendon Press.

Tarski, A., Mostowski, A., and Robinson, R., 1953. *Undecidable Theories.* Amsterdam: North-Holland Publishing Co.

Tourlakis, G., 2002. A programming formalism for PR. www.cs.yorku.ca/~gt/papers/loop-programs.ps.

Tourlakis, G., 2003. *Lectures in Logic and Set Theory.* Cambridge: Cambridge University Press.

Turing, A., 1936. On computable numbers, with an application to the *Entscheidungsproblem. Proceedings of the London Mathematical Society*, 42: 230–65. In Copeland 2004, pp. 58–90.

van Heijenoort, J. (ed.), 1967. *From Frege to Gödel.* Cambridge, MA: Harvard University Press.

von Neumann, J., 1927. Zur Hilbertschen Beweistheorie. *Mathematische Zeitschrift*, 26: 1–46.

Wittgenstein, L., 1989. *Wittgenstein's 1939 Cambridge Lectures on the Foundations of Mathematics.* C. Diamond, ed. Chicago: Chicago University Press.