An Introduction to

Gödel's Theorems

Peter Smith

Faculty of Philosophy
University of Cambridge

Version date: March 5, 2006

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

Contents

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 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 fine detail than you get in books for a more general audience (the best of those is Franzén 2005), but who find the rather stern style of 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 more accessible exposition.

But don't be misled by the relatively relaxed style and try to browse through too quickly. We do cover a reasonable amount of ground in quite a bit of detail, and new ideas often come thick and fast. Take things slowly!

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 of the key results we cover. However, my original plan shows through in two ways. First, some proofs are still only partially sketched in. 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. For those who want to fill in more details and test their understanding there will – in due course – be exercises on the book's website at www.godelbook.net.

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 – relatively elementary though certainly not always uncontentious – will also be reasonably widely accessible. Note

however that I am more interested in patterns of ideas and arguments than in being historically very precise when talking e.g. about logicism or about Hilbert's Programme.

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 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, Kin Tat Ko, 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 Luca Incurvati, Brian King, Mary Leng, and Hartley Slater who read carefully through a late draft chapters 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,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 its negation $\neg\varphi$ is deducible from the axioms, and so φ is false.[1] We may not, of course, actually stumble on a proof one way or the other: but such a proof always exists to be found, since the axioms contain enough information to enable the truth-value of any basic arithmetical proposition to 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 – we can in principle prove 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 they are bald definitions, or are logically derivable from definitions).

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

[1] '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.

flow deductively from logic plus definitions – then true arithmetical claims would be simply *analytic* in the philosophers' sense (i.e. would follow from logic plus definitions).[2] 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, *Gödel's First Incompleteness Theorem shows 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 derived 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 enrolled before than the second; a student's code-number starts with '1' she is an undergraduate student and with '2' if she is a graduate; and so on and so forth). In particular, then, we can use numbers and numerical propositions to encode facts about what can be proved in a theory T.[3] 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

[2] 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)

[3] It is absolutely standard for logicians to talk of a theory T as *proving* a sentence φ when there is a logically correct derivation of φ from T's assumptions. But T's assumptions may be contentious or plain false or downright absurd. So, T's proving φ in the logician's sense of course does not mean that φ is proved in the usual sense of established as true. It is far too late in the game to kick against the logician's usage, and in most contexts it is harmless. But our special concern in this book is the connections and contrasts between being true and being provable in this or that theory T. So we need to be on our guard. And to help emphasize that proving-in-T is not always proving-as-true, I'll often talk of 'deriving' rather than 'proving' sentences when it's the logician's notion which is in play.

arithmetical sentence G_T which encodes the claim 'The sentence G_T itself is unprovable in theory T'. So G_T is true if and only if T can't prove it.

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 G_T is true if and only if it is unprovable). Hence, if T is sound, G_T is unprovable in T. But then G_T is *true*. 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 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 a suitably constructed 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 nicely axiomatized 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 derive 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' D which records some basic facts about the course selections of a group of students – the language of D, 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 D 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 D 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, D 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 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. So we can iterate the same Gödelian construction

to generate a never-ending stream of independent true-but-unprovable sentences for any nicely axiomatized T including enough basic arithmetic.

1.3 More incompleteness

Incompletability doesn't just affect theories of 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.

The key point again: any axiomatized mathematical theory T rich enough to embed or model a reasonable amount of the basic arithmetic of the addition and multiplication of the natural numbers must be incomplete and incompletable

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

Now for another wonderful Gödelian insight. It turns out that the informal reasoning that we use, outside T, to show 'if T is consistent, then G_T is unprovable' is elementary enough to be mirrored by reasoning inside T (i.e. by

[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.

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 derive $\mathsf{Con}_T \rightarrow \mathsf{G}_T$ inside T.

But the First Theorem has already shown that if T is consistent we can't derive G_T in T. So it immediately follows that if T is consistent it can't prove Con_T. And *that* 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 that there's a genuine issue about whether T is consistent. Then even before we'd ever heard of Gödel's Second Theorem, we wouldn't have been convinced of its consistency by a derivation of Con_T inside T. For we'd just note that if T were in fact inconsistent, we'd be able to derive any T-sentence we like in the theory – including a statement of its own consistency!

The Second Theorem now shows that we would indeed be right not to trust a theory's announcement of its own consistency. For (assuming T includes enough arithmetic), if $T \vdash \mathsf{Con}_T$, then the theory must in fact be *in*consistent.

However, the real impact of the Second Theorem isn't in the limitations it places on a theory's proving its own consistency. The key point is this. 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). Hence we can't use nice 'safe' 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 internal 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 consistency proofs use 'safe' methods. 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. In Chapter 2, we introduce the notions of effective computability, decidability and enumerability, notions we are going to need in

[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, CCon_T, which also in some good sense expresses T's consistency, where T doesn't prove $\mathsf{CCon}_T \rightarrow \mathsf{G}_T$ but *does* prove CCon_T? We'll return to this question in Section **??**.

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 axiomatized theories in general. In Chapter 4, we need to introduce some concepts relating specifically to axiomatized theories of arithmetic (since Gödel's Theorems are about the limitations of such theories!). 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 some way short of Gödel's own 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.

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.

To use alternative jargon, these 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 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 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 ultra-idealized notions of computability and decidability. Why? Because later we'll be proving a range of limitative theorems, about what can't be algorithmically decided. And by working with a very 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 computable numerical functions and effectively decidable properties of numbers, and be very confident that we are referring to fully determinate classes (there isn't any relativity to computing architecture).

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 Xs.

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, **??**, **??**). 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. For each n put $g(2n) = f_1(n)$ and $g(2n+1) = f_2(n)$; then g enumerates $\Sigma_1 \cup \Sigma_2$.) ⊠

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:

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

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. ⊠

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-enumerably 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: a set is enumerable just if there exists some function or other that enumerates it. The function in question needn't be a 'nice' algorithmically computable one. 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] NB: whether a set is effectively enumerable, enumerable but not effectively so, or neither, depends on what functions there *are*, not on which functions we *know* about. Also note that terminology hereabouts isn't stable: some writers use 'enumerable' to mean *effectively* enumerable, and use e.g. 'denumerable' for the wider notion of enumerability.

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 for proofs which are not merely correct but *explanatory* – 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 formulae 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 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. Agreed, 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 $\mathcal{I}$ gives the intended interpretation of these expressions.

[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.

(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. It is useful to assume that our formal languages allow wffs with free variables (i.e. not all wffs are closed sentences).

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.

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.

(b) Let's move on, then, to the interpretation $\mathcal{I}$. Our prime aim is to fix the content of each $\mathcal{L}$-*sentence*, where a sentence is a closed wff without any unquantified variables left dangling free. 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 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 the expressions which it contains.

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,

> The value of 'a' is Socrates;
> the value of 'b' is Plato;
> something satisfies 'F' iff it is wise;

[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.).

> an ordered pair of things satisfies 'L' iff the first of them 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. Here's one way of telling the story. Intuitively, if the quantifier is to range over people, then '∃xFx' is true just if there is someone we could temporarily dub using 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 the quantifiers to run over (the set of people, for example). Then we can say that a sentence of the form $\exists\xi\varphi(\xi)$ is true on $\mathcal{I}$ just if we can expand the interpretation $\mathcal{I}$ by getting it to assign the new name κ a value in the domain, in such a way that $\varphi(\kappa)$ is true on the expanded interpretation.[6] Similarly for universal quantifiers.

For the moment, let's just make two comments about semantics. The first parallels our comment about syntax. Given the aims of formalization, a compositional semantics needs to yield an unambiguous truth-condition 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 it goes almost without saying: the usual accounts of the syntax and semantics of the standard formal languages of logic all have the decidability feature.

The second comment is this. Formal languages typically allow wffs which have free variables. Our semantic story so far assigns *them* no truth-conditions. And we could chose to treat them as indeed uninterpreted symbols that we use, e.g., in the course of inference games that take us from interpreted premisses to interpreted conclusions. But there's an alternative convention, which we'll

[6] 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.

adopt: treat wffs with free variables as true just when their *universal closures* are true (the universal closure of a wff is the result of prefixing the wff with a matching universal quantifier for each free variable – e.g. the universal closure of '(Lxy → Fx)' is '∀x∀y(Lxy → Fx)'). The rationale for this convention is that, if we have a deductive proof where wffs with free variables occur at various steps, then we'll still be able to say that if our initial axioms are true then the derived wff at each step of the proof is true.

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. Of course, we'll normally want axioms to be true on interpretation: but that isn't built into the very notion of an axiomatized theory.

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 derivation 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) Second, an axiomatized formal theory needs some deductive apparatus, i.e. some sort of formal *proof-system*. And we'll take proof derivations always to be *finite* arrays of wffs, arrays which are built up in ways that conform to the rules of the relevant proof-system.[7]

[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, there are proof systems including 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.

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 essentially matter. What is crucial, of course, 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 enables 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 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 derivation. This is *not* to say that we can always decide in advance whether a derivation from given premisses 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 **??**).

(c) Put together, then, the ideas that – in the case of an axiomatized formal theory T – it is decidable which wffs are T's axioms, and it is decidable which arrays of wffs obey the rules of T's proof-system. Then it will be decidable which arrays of wffs are axiomatic T-proofs (i.e. which arrays are properly constructed proofs, all of whose premisses are T-axioms).

To summarize then, here again are the key headlines:

> T is an (interpreted) axiomatized formal theory just if (i) 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 $\mathcal{L}$, and what the truth-condition of any sentence is, etc., (ii) it is effectively decidable which $\mathcal{L}$-wffs are axioms of T, and (iii) 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, and hence (iv) it is effectively decidable whether an array of $\mathcal{L}$-wffs counts as a proof from T's axioms.

[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).

3.4 More definitions

Here are five more definitions – absolutely standard ones – specifically to do with theories:

i. Given a derivation of 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 wff 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* 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 sentence 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 {'$\neg\mathsf{p}$'} and {'p', 'q', '$\neg\mathsf{r}$'}. 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 '$(\mathsf{q} \wedge \mathsf{r})$', since T_1's sole axiom doesn't entail either '$(\mathsf{q} \wedge \mathsf{r})$' or '$\neg(\mathsf{q} \wedge \mathsf{r})$'. 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]

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 derivations constructible in T, and (iii) 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) Assume that T-proofs are linear sequences of wffs. Just as we can enumerate all the possible wffs, so we can effectively enumerate all

[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 wffs Γ, and any φ, if $\Gamma \vDash \varphi$ then $\Gamma \vdash \varphi$,

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.

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 derivations from axioms of the theory. 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 possible derivations. (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 (iii) Start effectively enumerating the well-constructed derivations again. But this time, just record their conclusions. 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

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

(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 wff 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 can't be derived in T. 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 ...

$\forall x \forall y(x + y = y + x)$	$\forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} + \mathsf{y} = \mathsf{y} + \mathsf{x})$
$\exists y\ y = S0$	$\exists \mathsf{y}\ \mathsf{y} = \mathsf{S0}$
$1 + 2 = 3$	$1 + 2 = \mathsf{3}$

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 talked in Section 3.2 about sentences of the form $\exists\xi\varphi(\xi)$, or when we defined negation completeness in Section 3.4 by saying that for any sentence 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 recruit Greek letters like 'ξ' and 'ζ' to act as place-holders, indicating gaps to be filled in expressions. Note then that these symbols all 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 "¬" 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. Likewise, the extension of the numerical two-place relation R is the set of pairs of numbers m, n such that m is R to n. And we treat co-extensional relations as the same relation. (There's nothing at all 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 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 1.1. 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 'd', 'u' 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'.[1] We'll abbreviate the numerals $\mathsf{S0}$, $\mathsf{SS0}$, $\mathsf{SSS0}$, etc. by $\mathsf{1}$, $\mathsf{2}$, $\mathsf{3}$, etc.

[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.

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.)

Further, when we want to generalize, we'll write e.g. '$\overline{\mathsf{n}}$' to indicate the standard numeral $\mathsf{SS}\ldots\mathsf{S0}$ with n occurrences of 'S'. (Overlining is conventional, and helpfully distinguishes numerals from variables.)

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 (wffs with free variables are allowed).

(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, $val[\,\overline{\mathsf{n}}\,] = n$.

The atomic sentences (closed atomic 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 n in the intended domain *already* has a term to pick it

out, i.e. the numeral $\overline{\mathsf{n}}$. 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 $\exists\xi\varphi(\xi)$ (where 'ξ' can be any variable) is true iff, for some number n, $\varphi(\overline{\mathsf{n}})$ is true.

Similarly

> A sentence of the form $\forall\xi\varphi(\xi)$ is true iff, for any n, $\varphi(\overline{\mathsf{n}})$ 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 $(\mathsf{S0} + \mathsf{SS0}) = \mathsf{SSS0}$, is true just so long as one plus two is three; likewise $\exists\mathsf{x}\ 4 = (\mathsf{x} \times 2)$, i.e. $\exists\mathsf{x}\ \mathsf{SSSS0} = (\mathsf{x} \times \mathsf{SS0})$, 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

[2] 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. So for us his point becomes: claiming an L_A sentence φ is true commits us to nothing more than an assertion of a corresponding arithmetical sentence which renders φ into English. When we say that there are L_A truths which can't be proved in this or that formal system, the notion of truth is therefore not being used with any special weight.

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

the literal existence of such things. In the view of these philosophers, 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 our main concerns in this book. Fictionalists about arithmetic can systematically read our talk of various L_A sentences 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. $\psi(\overline{\mathsf{n}}) =_{\text{def}} \exists \mathsf{v}(2 \times \mathsf{v} = \overline{\mathsf{n}})$

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

> if n is even, then $\psi(\overline{\mathsf{n}})$ is true,
> if n isn't even, then $\neg\psi(\overline{\mathsf{n}})$ 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 $\psi(\overline{\mathsf{n}})$ 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*, at least in the sense of having 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. $\chi(\overline{\mathsf{n}}) =_{\text{def}} (\overline{\mathsf{n}} \neq 1 \ \wedge \ \forall \mathsf{u} \forall \mathsf{v}(\mathsf{u} \times \mathsf{v} = \overline{\mathsf{n}} \ \rightarrow \ (\mathsf{u} = 1 \ \vee \ \mathsf{v} = 1)))$

(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 $\chi(\overline{\mathsf{n}})$ is true,
> if n isn't prime, then $\neg\chi(\overline{\mathsf{n}})$ 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 $\varphi(\overline{\mathsf{n}})$ is true,[4]
>
> if n does not have the property P, then $\neg\varphi(\overline{\mathsf{n}})$ 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. $\psi(\overline{\mathsf{m}}, \overline{\mathsf{n}}) =_{\text{def}} \exists \mathsf{v}(\mathsf{v} + \overline{\mathsf{m}} = \overline{\mathsf{n}})$

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

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

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 $\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is true,
>
> if m does not have the relation R to n, then $\neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is true.

Likewise for many-place relations.[5]

[4] Do we need to spell it out? $\varphi(\overline{\mathsf{n}})$ is of course the result of substituting the numeral for n for each occurrence of 'x' in $\varphi(\mathsf{x})$.

[5] 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 reasons both technical and philosophical 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

(c) Let's emphasize again that 'expressing' in our sense is just a matter of getting the extension right. Suppose $\varphi(\mathsf{x})$ expresses the property P in L, and let θ be any true L-sentence. Then whenever $\varphi(\overline{\mathsf{n}})$ is true so is $\varphi(\overline{\mathsf{n}}) \wedge \theta$. And whenever $\varphi(\overline{\mathsf{n}})$ is false so is $\varphi(\overline{\mathsf{n}}) \wedge \theta$. Which means that $\varphi'(\mathsf{x}) =_{\text{def}} \varphi(\mathsf{x}) \wedge \theta$ also expresses P – irrespective of what θ means.

Hence, we might say, $\varphi(\mathsf{x})$'s expressing P in our sense is just a necessary condition for its expressing that property in the more intuitive sense of having the right meaning. But the intuitive notion is murky and notoriously difficult to analyse; our nice, sharply defined, notion will in fact serve us perfectly well for nearly every purpose.

4.5 Capturing numerical properties and relations

(a) 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 (more carefully: we want formal derivations which will be proofs in the intuitive sense if we take it that the axioms are established truths).

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 $T \vdash \varphi(\overline{\mathsf{n}})$,
>
> if n does not have the property P, then $T \vdash \neg\varphi(\overline{\mathsf{n}})$.

to mark the gaps; and sticking to this practice therefore 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 10.3, 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 implicit 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.

For example, in theories of arithmetic T with very modest axioms, the wff $\psi(\mathsf{x}) =_{\text{def}} \exists\mathsf{v}(2 \times \mathsf{v} = \mathsf{x})$ not only expresses but captures the property *even*. In other words, for each even n, T can prove $\psi(\overline{\mathsf{n}})$, and for each odd n, T can prove $\neg\psi(\overline{\mathsf{n}})$. 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 $T \vdash \varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$
> if m does not have the relation R to n, then $T \vdash \neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$.

Likewise for many-place relations.

(b) We should add a comment which parallels the point we made about 'expressing'. Suppose $\varphi(\mathsf{x})$ captures the property P in T, and let θ be any T-theorem. Then whenever $T \vdash \varphi(\overline{\mathsf{n}})$, then $T \vdash \varphi(\overline{\mathsf{n}}) \wedge \theta$. And whenever $T \vdash \neg\varphi(\overline{\mathsf{n}})$, then $T \vdash \neg(\varphi(\overline{\mathsf{n}}) \wedge \theta)$. Which means that $\varphi(\mathsf{x}) =_{\text{def}} \varphi(\mathsf{x}) \wedge \theta$ also captures P – irrespective of θ's content.

Hence, we might say, $\varphi(\mathsf{x})$'s capturing P in our sense is just a necessary condition for its capturing that property in a more intuitive sense of being able to prove wffs with the right meaning to be just about P. But again the intuitive notion is too murky, and our sharply defined notion will serve us perfectly well for nearly every purpose.

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 (see Chapter 10). But there are minor complications 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*.[6]

[6] 'Expresses' is used in our way by e.g. Smullyan (1992, p. 19). As alternatives, we find e.g.

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 **??**). 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 $T \vdash \varphi(\overline{\mathsf{n}})$, then $\varphi(\overline{\mathsf{n}})$ is true. And if $T \vdash \neg\varphi(\overline{\mathsf{n}})$, then $\neg\varphi(\overline{\mathsf{n}})$ is true. Which entails that *if* $\varphi(\mathsf{x})$ *captures* P *in the sound theory* T *then, a fortiori,* $\varphi(\mathsf{x})$ *expresses* P.

'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' – (Smoryński, 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, $T \vdash \varphi(\overline{\mathsf{n}})$, and if n does not have property P, $T \vdash \neg\varphi(\overline{\mathsf{n}})$ – 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 wffs 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 mechanical rules for determining whether a variable is free or bound).

Now let's fix on the following definition:

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

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

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

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

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

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

From (ii) and (iii), it follows that whether d has property D or not, the wff $\neg\varphi_d(\overline{\mathsf{d}})$ is a theorem either way. So by (iii) again, d does have property D, hence by (i) the wff $\varphi_d(\overline{\mathsf{d}})$ 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 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 of T.

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

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

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

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 in 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 $Prf(m, n)$ 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 **??**.)

4. Next – the really exciting bit! – we use the fact that relations like *Prf* are expressible in PA to construct a 'Gödel sentence' G. Given the coding scheme, G will be true when 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 G is true just if it is 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. (Chapter **??**)

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′; throwing in G′ as a further axiom gives us another theory for we can generate yet another true-but-unprovable wff. PA is therefore not only incomplete but incompletable. Indeed, *any* properly axiomatized theory that contains the weaker theory Q is incompletable. (Chapter **??**)

Just one comment. On the face of it, this story makes Gödel's proof sounds very different from our informal proof using the idea of 'sufficiently strong' theories which capture all decidable properties. But a close connection can be made when we note that the primitive recursive properties are a large and important sub-class of the intuitively decidable properties. Hence showing that Q and PA can capture all primitive recursive properties is in fact 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 a number of particular formal theories of arithmetic. In this chapter, we limber up by looking at the trivial theory Baby Arithmetic and then start exploring the pivotal Robinson Arithmetic. And then in Chapter 8 we work up to considering Peano Arithmetic (the strongest of our various first-order arithmetics). 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 mention 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. 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 *schemata* and say: *any sentence that you get from one of the following schemata 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 schemata do indeed entail that different terms in the sequence $\mathsf{0}$, $\mathsf{S0}$, $\mathsf{SS0}$, $\mathsf{SSS0}$, ..., pick out different numbers. Recall, we use '$\overline{\mathsf{n}}$' 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 $\mathsf{BA} \vdash \overline{\mathsf{m}} \neq \overline{\mathsf{n}}$.

Proof Suppose $m \neq n$ (and let $|m-n|-1 = j \geq 0$). And assume $\overline{\mathsf{m}} = \overline{\mathsf{n}}$ 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 $\mathsf{0} = \mathsf{S}\bar{\mathsf{j}}$, or else (ii) we derive $\mathsf{S}\bar{\mathsf{j}} = \mathsf{0}$ and then use the symmetry of identity to conclude $\mathsf{0} = \mathsf{S}\bar{\mathsf{j}}$. But $\mathsf{0} \neq \mathsf{S}\bar{\mathsf{j}}$ is an axiom (an instance of Schema 1). Contradiction. So, $\overline{\mathsf{m}} \neq \overline{\mathsf{n}}$ follows by reductio. Which proves that if $m \neq n$, then $\mathsf{BA} \vdash \overline{\mathsf{m}} \neq \overline{\mathsf{n}}$. ⊠

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 $\zeta \times 0 = 0$

Schema 6 $\zeta \times S\xi = (\zeta \times \xi) + \zeta$

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 6 with 'ξ' replaced by 'S0' to tell us how to multiply by two in terms of multiplying by one and doing some addition. 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 Schemata, 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 derive 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 $SS0 \times S0 = SS0$.

1.	$SS0 \times 0 = 0$	Instance of Schema 5
2.	$SS0 \times S0 = (SS0 \times 0) + SS0$	Instance of Schema 6
3.	$SS0 \times S0 = 0 + SS0$	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 $0 + SS0 = SS0$ – and note, this *isn't* an instance of Schema 3. So

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

Which gives us what we want:

9.	$SS0 \times S0 = SS0$	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 $\bar{\mathsf{j}} + \bar{\mathsf{k}} = \overline{\mathsf{m}}$ or $\bar{\mathsf{j}} \times \bar{\mathsf{k}} = \overline{\mathsf{n}}$. 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 $\tau = \bar{\mathsf{t}}$ with a numeral on the right by repeated steps of evaluating inner-most brackets.

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

$$\mathsf{SS}((\bar{\mathsf{j}} + \bar{\mathsf{k}}) + \mathsf{S}(\bar{\mathsf{j}} \times \bar{\mathsf{k}})) = \mathsf{SS}(\overline{\mathsf{m}} + \mathsf{S}\overline{\mathsf{n}})$$

Now evaluate the new, simpler, bracketed expression on the right by proving something of the form $\overline{\mathsf{m}} + \mathsf{S}\overline{\mathsf{n}} = \overline{\mathsf{o}}$. Hence, using Leibniz's Law again, we get

$$\mathsf{SS}((\bar{\mathsf{j}} + \bar{\mathsf{k}}) + \mathsf{S}(\bar{\mathsf{j}} \times \bar{\mathsf{k}})) = \mathsf{SS}\overline{\mathsf{o}}$$

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 derive a pair of wffs of the form $\tau = \overline{\mathsf{m}}$, $\rho = \overline{\mathsf{n}}$. But we've already shown in the previous section that if $m \neq n$, BA proves $\overline{\mathsf{m}} \neq \overline{\mathsf{n}}$. 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 will be some version of first-order logic with identity. In the next chapter, we'll fix on a convenient official logic.

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 Schemata governing the successor function by

Axiom 1 $\forall x(0 \neq Sx)$

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

Each instance of our earlier Schemata 1 and 2 can be deduced from the corresponding Axiom.

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 $\forall x(x \neq 0 \rightarrow \exists y(x = Sy))$

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

Axiom 4 $\forall x(x + 0 = x)$

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

Axiom 6 $\forall x(x \times 0 = 0)$

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

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

Q is a sound theory. Its axioms are all true; its logic is truth-preserving; so its derivations are proper proofs in the intuitive sense of demonstrations of truth, and every theorem of Q is true. But just which truths are theorems?

Since any BA Axiom – i.e. any instance of one of our previous Schemata – 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 $0 + \overline{n} = \overline{n}$. But it can't prove the universal generalization $\chi =_{\text{def}} \forall x(0 + x = x)$.

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, re-interpretation) for the T-wffs which makes the axioms of T true and hence all its theorems true, but which makes χ false.

So take $L_A = \langle \mathcal{L}_A, \mathcal{I}_A \rangle$, the interpreted language of Q. What we want to find is a deviant re-interpretation $\mathcal{I}_D$ of the same wffs $\mathcal{L}_A$, where $\mathcal{I}_D$ still makes Q's Axioms true but allows cases where 'adding' a 'number' to zero changes it. Here's an artificial – but still legitimate – example.

Take the domain of our deviant, unintended, interpretation $\mathcal{I}_D$ 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 interpretation $\mathcal{I}_D$ to re-interpret Q's function '+'. Suppose we take this to pick out addition*, where $m +^* n = m + n$ for any

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

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 '$\times$' in Q by some deviant multiplication* function. But we can leave it as an exercise to fill in suitable details. ⊠

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 elaborate 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 $\forall x(0 + x = x)$, Q is evidently a *very* weak theory of arithmetic. Even so, despite its great shortcomings, Q does have some really nice properties which make it of special interest. In particular, and perhaps rather surprisingly, it turns out that Q is *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 it turns out that Q's hopeless weakness at proving generalizations doesn't stop it doing that.

Establishing the full claim about sufficient strength is business for much later (plainly, we can only establish it when we have a general theory of decidability to hand). But we'll be able to prove some more modest, but still very interesting, claims about Q along the way. Indeed, before we think about how to beef up Q into a theory that is more competent to deal with elementary generalizations, we'll pause in the next chapter to have a first look at some things that 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

Q is a weak theory: but we'll explore here what it *can* establish. Unavoidably, some of the detailed arguments do get boringly fiddly; and so you are very welcome to skip many of the *proofs* of stated results (you won't miss anything exciting). However, you will need to carry forward to later chapters a good understanding of the key *concepts* we'll be introducing.

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

1. We begin with some quick preliminaries about different systems of first-order logic.
2. We next 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)).
3. This motivates our adding the symbol '$\leq$' to Q so that $\xi \leq \zeta$ is a definitional abbreviation of $\exists \mathsf{v}(\mathsf{v} + \xi = \zeta)$.
4. We then show that Q can formally prove a range of expected results about the less-than-or-equals-to relation.
5. Next we introduce the class of so-called Δ_0 wffs: these are *bounded* L_A wffs, i.e. wffs which are built up using identity, the less-than-or-equals relation, propositional connectives and bounded quantifiers. We also introduce the class of Σ_1 wffs which are (equivalent to) unbounded existential quantifications of Δ_0 wffs, and the Π_1 wffs which are (equivalent to) unbounded universal quantifications of Δ_0 wffs.
6. Consider a *bounded* existential quantification, as in e.g. $(\exists \mathsf{x} \leq \overline{\mathsf{n}})\varphi(\mathsf{x})$ when read in the obvious way. That quantification is equivalent to the finite disjunction $\varphi(\mathsf{0}) \vee \varphi(\mathsf{1}) \vee \ldots \vee \varphi(\overline{\mathsf{n}})$. Likewise a *bounded* universal quantification $(\forall \mathsf{x} \leq \overline{\mathsf{n}})\varphi(\mathsf{x})$ is equivalent to a finite conjunction. Δ_0 *sentences* that are built up with such bounded quantifiers are therefore like the quantifier-free sentences to which they are equivalent. It will be a simple matter of mechanical calculation to determine whether they are true or not.
7. We then show that Q knows enough about bounded quantifiers to be able to correctly decide every bounded sentence – i.e. prove it if it is true, disprove it if it is false. And that quickly gives us the key theorem that Q can also prove any true existentially quantified bounded wff (in the jargon, can prove any true Σ_1 sentence).
8. We finally note an intriguing corollary of that last result.

7.1 Systems of logic

As will be very familiar, there is a wide variety of formal deductive systems for first-order logic (systems which are equivalent in the sense of proving at least the same closed wffs from given premisses). Let's contrast, in particular, Hilbert-style axiomatic systems with varieties of natural deduction system.

A Hilbert-style system defines a class of logical axioms, usually by giving schemata such as $(\varphi \rightarrow (\psi \rightarrow \varphi))$ and $(\forall\xi\varphi(\xi) \rightarrow \varphi(\tau))$ and then stipulating that – with some restrictions – any instance of a schema is an axiom. Having a very rich set of axioms, such a deductive system can make do with just one or two rules of inference. And a proof in a theory using an axiomatic logic is then a linear sequence of wffs, each one of which is either (i) a logical axiom, or (ii) an axiom belonging to the specific theory, or (iii) follows from previous wffs in the sequence by one of the rules of inference.[1]

A natural deduction system, on the other hand, will have no logical axioms but many rules of inference. And, in contrast to axiomatic logics, we allow temporary assumptions to be made for the sake of argument and then later discharged. We will need some way, therefore, of keeping track of which temporary assumptions are in play when. So it becomes compelling to set out proofs as non-linear arrays. One option is to use tree structures of the type Gerhard Gentzen introduced. Another option is to use Frederic Fitch's device of indenting a column of argument to the right each time a new assumption is made, and shifting back to the left when the assumption is discharged.[2]

Which style of logical system should we adopt in developing Q and other arithmetics with a first-order logic? Well, that will depend e.g. on whether we are more concerned with the ease of proving certain metalogical results *about* formal arithmetics or with the ease of proving results *inside* the theories. Hilbertian systems are very amenable to metalogical treatment but are horrible to use in practice. Natural deduction systems are indeed natural in use; but it takes more effort to theorize about arboriform proofs structures.

I propose that in this book we cheerfully have our cake and eat it. So when we consider arithmetics like Q, then *officially* we'll take their logic to be a Hilbertian, axiomatic one, so that proofs are linear sequences. This way, when we come to theorize *about* arithmetic proofs, and e.g. use the Gödelian trick of using numbers to code proofs, everything goes as simply as it can. However, when we want to give sample proofs *inside* formal arithmetics, as in the next section, we will outline arguments framed in a more manageable natural deduction style. The familiar equivalences between the different logical systems will then warrant the implication that an official Hilbert-style proof of the same result will be available.

[1] The *locus classicus* is (Hilbert and Ackermann, 1928). For a modern logic text which uses a Hilbertian system, see (Mendelson, 1997).

[2] The *locus classicus* for natural deduction systems is, of course, (Gentzen, 1935). For a modern text which uses a natural deduction system set out in tree form, see e.g. (van Dalen, 1994). Frederic Fitch introduces his elegant way of setting out proofs in his (1952).

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

In this section, then, we'll show that the *less-than-or-equal-to* relation is captured by the wff $\exists v(v + x = y)$ in Q. That is to say, for any particular pair of numbers, m, n, if $m \leq n$, then $Q \vdash \overline{m} \leq \overline{n}$, and otherwise $Q \vdash \neg\overline{m} \leq \overline{n}$.

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 $Q \vdash \overline{k} + \overline{m} = \overline{n}$. But $\overline{k} + \overline{m} = \overline{n} \vdash \exists v(v + \overline{m} = \overline{n})$ by existential quantifier introduction. Therefore $Q \vdash \exists v(v + \overline{m} = \overline{n})$, as was to be shown.

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

Because it is well known (but also simple to follow if you don't know it), we will use a Fitch-style system. As we noted before, we indent sub-proofs while a new temporary assumption is in force. And we use symbols to act as temporary names ('parameters'): you can think of these as recruited from our stock of unused variables.

So consider the following argument (for brevity we will omit statements of Q's axioms, and some other trivial steps; LL of course indicates the use of Leibniz's Law):

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

The only step to explain is at line (14) where we use a version of Existential Elimination: the principle invoked is that if the supposition $\varphi(a)$ leads to contradiction, for arbitrary a, then $\exists v\varphi(v)$ also leads to contradiction.

And inspection of our proof for this one particular case immediately reveals that we can use the same pattern of argument to show that $Q \vdash \neg\exists v(v + \overline{m} = \overline{n})$ whenever $m > n$. (Exercise: check that!) So we are done. ⊠

7.3 Adding '$\leq$' to L_A

Given the result we've just proved, it is evidently appropriate now to add the standard symbol '$\leq$' to L_A, defined so that (whatever we put for 'ξ' and 'ζ'), $\xi \leq \zeta =_{\text{def}} \exists \mathsf{v}(\mathsf{v} + \xi = \zeta)$.[3] Since it greatly helps readability, we'll henceforth make very free use of this abbreviatory symbol inside formal arithmetics.

Let's also adopt a second, closely related, abbreviatory convention. In informal mathematics we standardly express *bounded quantifications* – which say that all/some numbers less than or equal to a given number have some particular property – by expressions like $(\forall x \leq n)Fx$ and $(\exists x \leq n)Fx$. We can express such claims formally by wffs of the shape $\forall\xi(\xi \leq \kappa \rightarrow \varphi(\xi))$ and $\exists\xi(\xi \leq \kappa \wedge \varphi(\xi))$. It is standard to abbreviate such formal wffs by $(\forall\xi \leq \kappa)\varphi(\xi)$ and $(\exists\xi \leq \kappa)\varphi(\xi)$ respectively. By co-ordinating our informal and formal modes of expressions, this extra convention again aids readability.

7.4 Ten simple facts about what Q can prove

We could now go on to show, example by example, that various other particular 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-built to be about the weakest tidy theory that has this nice property.

In this section, then, we'll instead note some basic *general* facts about what we can derive inside Q (chosen with an eye to what we will eventually need for our proof that Q is indeed 'sufficiently strong'). So here are ten:

1. $\mathsf{Q} \vdash \forall\mathsf{x}(0 \leq \mathsf{x})$.
2. For any n, $\mathsf{Q} \vdash \forall\mathsf{x}(\mathsf{Sx} + \overline{\mathsf{n}} = \mathsf{x} + \mathsf{S}\overline{\mathsf{n}})$.
3. For any n, $\mathsf{Q} \vdash \forall\mathsf{x}(\{\mathsf{x} = 0 \vee \mathsf{x} = 1 \vee \ldots \vee \mathsf{x} = \overline{\mathsf{n}}\} \rightarrow \mathsf{x} \leq \overline{\mathsf{n}})$.
4. For any n, $\mathsf{Q} \vdash \forall\mathsf{x}(\mathsf{x} \leq \overline{\mathsf{n}} \rightarrow \{\mathsf{x} = 0 \vee \mathsf{x} = 1 \vee \ldots \vee \mathsf{x} = \overline{\mathsf{n}}\})$.
5. For any n, if $\mathsf{Q} \vdash \varphi(0)$, $\mathsf{Q} \vdash \varphi(1)$, $\ldots$, $\mathsf{Q} \vdash \varphi(\overline{\mathsf{n}})$, then $\mathsf{Q} \vdash (\forall\mathsf{x} \leq \overline{\mathsf{n}})\varphi(\mathsf{x})$.

[3] 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. We should also remark, by the way, that some presentations treat '$\leq$' 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 '$\leq$' as our basic symbol rather than '$<$' (which could have been defined by $\xi < \zeta =_{\text{def}} \exists\mathsf{v}(\mathsf{Sv} + \xi = \zeta)$.)

6. For any n, if $\mathsf{Q} \vdash \varphi(0)$, or $\mathsf{Q} \vdash \varphi(1)$, ..., or $\mathsf{Q} \vdash \varphi(\overline{\mathsf{n}})$, then $\mathsf{Q} \vdash (\exists \mathsf{x} \leq \overline{\mathsf{n}})\varphi(\mathsf{x})$.
7. For any n, $\mathsf{Q} \vdash \forall \mathsf{x}(\mathsf{x} \leq \overline{\mathsf{n}} \rightarrow \mathsf{x} \leq \mathsf{S}\overline{\mathsf{n}})$.
8. For any n, $\mathsf{Q} \vdash \forall \mathsf{x}(\overline{\mathsf{n}} \leq \mathsf{x} \rightarrow (\overline{\mathsf{n}} = \mathsf{x} \vee \mathsf{S}\overline{\mathsf{n}} \leq \mathsf{x}))$.
9. For any n, $\mathsf{Q} \vdash \forall \mathsf{x}(\mathsf{x} \leq \overline{\mathsf{n}} \vee \overline{\mathsf{n}} \leq \mathsf{x})$.
10. For any $n > 0$, $\mathsf{Q} \vdash (\forall \mathsf{x} \leq \overline{\mathsf{n-1}})\varphi(\mathsf{x}) \rightarrow (\forall \mathsf{x} \leq \overline{\mathsf{n}})(\mathsf{x} \neq \overline{\mathsf{n}} \rightarrow \varphi(\mathsf{x}))$

These belong squarely in the class of 'trivial but tiresome' results. In other words, they are (or at least, some of them are) a bit fiddly to demonstrate: but none of the proofs involves anything deep. If you have a taste for logical brain-teasers, then see how many of these results you can establish before reading on. If you don't, just make sure that you understand the content of these results and then feel entirely free to jump on to the next section.[4]

Still reading? Then as a preliminary, let's clarify notation. We are using '$\overline{\mathsf{n}}$' to indicate L_A's standard numeral for n. In this notation, then, '$\overline{\mathsf{n}} + 1$' indicates the expression you get by writing the numeral for n followed by a plus sign followed by '1'. What, then, if we want to indicate instead the numeral for $n+1$? That's '$\overline{\mathsf{n+1}}$'. The scope of the overlining shows what is being wrapped up into a single numeral. (So contrast: '$\overline{\mathsf{n-1}}$' stands in for the numeral for $n-1$; while '$\overline{\mathsf{n}} - 1$' is ill-formed since '$-$' is not a symbol of L_A.)

Proof for (1) For arbitrary a, Q proves $\mathsf{a} + 0 = \mathsf{a}$, hence $\exists \mathsf{v}(\mathsf{v} + 0 = \mathsf{a})$, i.e. $0 \leq \mathsf{a}$. Generalize to get the desired result. ⊠

Proof for (2) The following holds in Q for arbitrary a:

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

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 again generalize. ⊠

Proof for (3) Arguing in Q, suppose that $\mathsf{a} = 0 \vee \mathsf{a} = 1 \vee \ldots \vee \mathsf{a} = \overline{\mathsf{n}}$. We showed in Section 7.2 that if $k \leq m$, then Q proves $\overline{\mathsf{k}} \leq \overline{\mathsf{m}}$. Which means that from each disjunct we can derive $\mathsf{a} \leq \overline{\mathsf{n}}$. Hence, arguing by cases, $\mathsf{a} \leq \overline{\mathsf{n}}$. So, discharging the supposition, Q proves $\mathsf{a} = 0 \vee \mathsf{a} = 1 \vee \ldots \vee \mathsf{a} = \overline{\mathsf{n}} \rightarrow \mathsf{a} \leq \overline{\mathsf{n}}$. The desired result is immediate since a was arbitrary. ⊠

Proof for (4) 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

[4] '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 the art of reading mathematical texts!

(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.[5]

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

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

Our supposition, unpacked, is $\exists v(v + a = \overline{k+1})$. And by Axiom 3, (i) $a = 0$ or (ii) $a = Sa^\circ$, for some a°.

Exploring case (ii), we then have $\exists v(v + Sa^\circ = S\overline{k})$: hence, by Axiom 5 and Axiom 2, we can derive $\exists v(v + a^\circ = \overline{k})$; i.e. $a^\circ \leq \overline{k}$. So, using our given assumption that the result holds for k, $a^\circ = 0 \vee a^\circ = 1 \vee \ldots \vee a^\circ = \overline{k}$. Since $a = Sa^\circ$, that implies $a = 1 \vee a = 2 \vee \ldots \vee a = \overline{k+1}$.

Hence, putting (i) and (ii) together, $a = 0 \vee a = 1 \vee a = 2 \vee \ldots \vee a = \overline{k+1}$. ⊠

Proof for (5) Assume Q proves each of $\varphi(0)$, $\varphi(1)$, …, $\varphi(\overline{n})$. Then, suppose $a \leq \overline{n}$, for arbitrary a. By Result 4, we have $a = 0 \vee a = 1 \vee \ldots \vee a = \overline{n}$. Now we argue by cases: each disjunct separately – combined with one of our initial suppositions – gives $\varphi(a)$; so we can conclude $\varphi(a)$. Discharging our temporary supposition, $a \leq \overline{n} \rightarrow \varphi(a)$. Generalize, and we are done. ⊠

Proof for (6) Assume Q proves $\varphi(\overline{k})$ for some $k \leq n$. Since Q captures *less-than-or-equal-to*, Q proves $\overline{k} \leq \overline{n} \wedge \varphi(\overline{k})$, hence $\exists x(x \leq \overline{n} \wedge \varphi(x))$. ⊠

Proof for (7) Suppose $a \leq \overline{n}$ for arbitrary a. Then $a = 0 \vee a = 1 \vee \ldots \vee a = \overline{n}$ by Result 4. So by trivial logic, $a = 0 \vee a = 1 \vee \ldots \vee a = \overline{n} \vee a = \overline{n+1}$. So by Result 3, $a \leq \overline{n+1}$, i.e. $a \leq S\overline{n}$. Which gives us what we want. ⊠

Proof for (8) Suppose $\overline{n} \leq a$ for arbitrary a. So for some b, $b + \overline{n} = a$. By Axiom 3, either (i) $b = 0$ or (ii) $b = Sb^\circ$ for some b°. If (i), then $0 + SS\ldots S0 = a$, and applying Axiom 5 n times and then Axiom 4 gives us $\overline{n} = a$. If (ii), $Sb^\circ + \overline{n} = a$, so by Result 2 we have $b^\circ + S\overline{n} = a$, hence $\exists v(v + S\overline{n} = a)$, i.e. $S\overline{n} \leq a$. Therefore either way we get $\overline{n} = a \vee S\overline{n} \leq a$, and we are done. ⊠

Proof for (9) We show that (a) the given wff is provable in Q 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.

[5] 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.

(a) We saw in proving Result 1 that for any a, $0 \leq \mathsf{a}$. A fortiori, $0 \leq \mathsf{a} \vee \mathsf{a} \leq 0$. 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.	$\mathsf{a} \leq \overline{\mathsf{k}} \vee \overline{\mathsf{k}} \leq \mathsf{a}$	By our supposition
ii.	$\mathsf{a} \leq \overline{\mathsf{k}} \rightarrow \mathsf{a} \leq \overline{\mathsf{k+1}}$	By Result 7
iii.	$\overline{\mathsf{k}} \leq \mathsf{a} \rightarrow \overline{\mathsf{k}} = \mathsf{a} \vee \overline{\mathsf{k+1}} \leq \mathsf{a}$	By Result 8

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

iv.	$\mathsf{a} = \overline{\mathsf{k}} \rightarrow \mathsf{a} \leq \overline{\mathsf{k+1}}$	
v.	$\mathsf{a} \leq \overline{\mathsf{k+1}} \vee \overline{\mathsf{k+1}} \leq \mathsf{a}$	From (i) to (iv)

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

Proof for (10) To finish, an easy result which we list for future use. Arguing inside Q, suppose $(\forall \mathsf{x} \leq \overline{\mathsf{n-1}})\varphi(\mathsf{x})$. By Result 4, we have, $\varphi(0) \wedge \varphi(1) \wedge \ldots \ldots \wedge \varphi(\overline{\mathsf{n-1}})$. We can trivially prove $\overline{\mathsf{n}} \neq \overline{\mathsf{n}}$. So we can derive $(0 \neq \overline{\mathsf{n}} \rightarrow \varphi(0)) \wedge (1 \neq \overline{\mathsf{n}} \rightarrow \varphi(1)) \wedge \ldots \wedge (\overline{\mathsf{n-1}} \neq \overline{\mathsf{n}} \rightarrow \varphi(\overline{\mathsf{n-1}})) \wedge (\overline{\mathsf{n}} \neq \overline{\mathsf{n}} \rightarrow \varphi(\overline{\mathsf{n}}))$. Finally, by Result 5, that entails $(\forall \mathsf{x} \leq \overline{\mathsf{n}})(\mathsf{x} \neq \overline{\mathsf{n}} \rightarrow \varphi(\mathsf{x}))$. ⊠

7.5 Defining the Δ_0, Σ_1 and Π_1 wffs

If you skipped those proofs, that's fine: but now concentrate again because the concepts defined in this section are crucial!

(a) Why did we bother to prove all those 'trivial but tiresome' results? Because they enable us easily to show that Q can prove *every* true sentence in the class of so-called Σ_1 wffs.

And why do we care about this class of wffs (whatever it is)? Because later we'll also be able to show that it contains 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 hence is sufficiently strong.

So what, then, are these Σ_1 wffs? *They are – or are equivalent to – wffs which begin with one or more ordinary, unbounded,* existential *quantifiers followed up by a* bounded *kernel wff.*

And a bounded wff – a Δ_0 wff, to use the standard jargon – 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.[6] As we'll soon see, just as Q can prove all true equations and

[6] 'Hold on! What exactly 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

disprove all false ones, it can also correctly decide all Δ_0 sentences 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 sentences.

(b) So much for the brisk headline news. Next, the official definitions to fill in the details. We'll 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.[7]

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.[8]
4) Nothing else is a Δ_0 wff.

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.

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.

(c) A comment. It is worth remarking that 'Σ' in the standard label 'Σ_1' comes from an old alternative notation for the existential quantifier, as in ΣxFx (a Greek 'S' for 'sum'). Likewise the 'Π' in 'Π_1' comes from corresponding notation for the universal quantifier, as in ΠxFx (a Greek 'P' for 'product'). 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. (By the same token, a Σ_0 wff will be one in which a bounded kernel is preceded by no block of quantifiers – and therefore the Σ_0 wffs are just the Δ_0 wffs, as we've dubbed them. In fact both labels are current.[9])

(d) Some examples might help:

---

are on a par.' True enough! So let's be more careful and say that an existential quantifier, say $\exists$x, has a *bounded occurrence* when it occurs in a subformula of the type $\exists$x(x $\leq \kappa \wedge \varphi$(x)), for some numeral or variable κ (other than 'x'). Similarly, a universal quantifier, say $\forall$x, has a bounded occurrence when it occurs in a subformula of the form $\forall$x(x $\leq \kappa \rightarrow \varphi$(x)). Then the idea will be that a bounded wff, if it involves quantifiers at all, involves only quantifiers with bounded occurrences.

[7] A quick reminder: 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)).

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

[9] Note, the general concepts of Δ_0, Σ_1 and Π_1 wffs are absolutely standard; however, the initial definitions given to these concepts do vary in a number of ways across different pre-

1. $\mathsf{S0} \leq \mathsf{SSS0}$, $\mathsf{x} = \mathsf{SSS0} + \mathsf{y}$, and $\mathsf{SS}(\mathsf{y} \times \mathsf{S0}) = \mathsf{y} \times \mathsf{SSy}$ are atomic Δ_0 wffs.
2. $(\exists \mathsf{x} \leq \mathsf{SS0})\, \mathsf{x} \neq \mathsf{0}$, $(\exists \mathsf{y} \leq \mathsf{SSS0})\, \mathsf{x} = \mathsf{S0} + \mathsf{y}$, $\neg(\exists \mathsf{z} \leq \mathsf{SSSSS0})(\forall \mathsf{y} \leq \mathsf{x})\, \mathsf{y} \leq \mathsf{z}$, and $((\exists \mathsf{x} \leq \mathsf{y})\, \mathsf{2} \times \mathsf{x} = \mathsf{y} \ \vee \ \mathsf{y} = \mathsf{0})$ are Δ_0 wffs.
3. $\exists \mathsf{x} \exists \mathsf{y}\, \mathsf{y} + \mathsf{S0} = \mathsf{x}$ and $\exists \mathsf{x}(\exists \mathsf{y} \leq \mathsf{x})\, \mathsf{y} + \mathsf{SSS0} = \mathsf{Sx}$ are strictly Σ_1 wffs.
4. $\neg \forall \mathsf{x} \forall \mathsf{y}\, \mathsf{y} + \mathsf{S0} \neq \mathsf{x}$ and $\neg \forall \mathsf{x}(\forall \mathsf{y} \leq \mathsf{x})\, \mathsf{y} + \mathsf{SSS0} \neq \mathsf{Sx}$ are Σ_1 wffs since they are logically equivalent to our sample strictly Σ_1 wffs.
5. $\exists \mathsf{x}(\mathsf{S0} \times \mathsf{0} = \mathsf{x} \wedge \mathsf{x} = \mathsf{0})$ is Σ_1. But that wff is equivalent to $\mathsf{S0} \times \mathsf{0} = \mathsf{0}$, which is equivalent to $\forall \mathsf{x}(\mathsf{S0} \times \mathsf{0} = \mathsf{x} \leftrightarrow \mathsf{x} = \mathsf{0})$, which is strictly Π_1. So the first wff is also Π_1. Hence being Σ_1 and being Π_1 are *not* exclusive – though, of course, being *strictly* Σ_1 trivially excludes being *strictly* Π_1.

7.6 Some easy results

(a) Three mini results to write into the record:

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.

Proof The first is trivial. The second is also trivial given the familiar equivalence of '$\neg\exists$x' with '$\forall$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. ⊠

(b) Another basic result: *we can work out the truth or falsity of any closed* Δ_0 *wff by a simple mechanical calculation*, basically because we only have to look through a finite number of cases when we have to deal with a *bounded* quantification.

Proof You might think our claim is too obvious really to need a proof. Fair enough. But we'll give the argument all the same (though skip if you must!) – for it nicely illustrates a standard technique, namely proving a result about all wffs in some class by an *induction on the complexity of the wff*.[10]

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.

sentations. Don't be fazed by this. For, given enough background setting, the various versions are equivalents, and the choice of an initial definition is largely a matter of convenience. For example, it in fact mostly doesn't matter if we alternatively define a Σ_1 wff as one which starts with a *single* existential quantifier before its Δ_0 kernel (for a proof, see Section **??**.)

[10] For the principle underlying inductive arguments, see Section 8.1.

The degree 0 sentences are the atomic Δ_0 sentences; and we can obviously calculate the truth-value of any such sentence.

Suppose then that we can mechanically calculate the truth-value of any Δ_0 sentence of degree no more than k. Then we can calculate the truth-value of any degree $k+1$ sentence χ too. For there are just three sorts of case to consider:

i. χ is of the form $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$ or $(\varphi \leftrightarrow \psi)$, where φ and ψ are Δ_0 sentences of degree no greater than k. So the truth-value of any relevant φ and ψ is by hypothesis mechanically calculable, and hence (using truth-tables) so is the truth value of χ.

ii. χ is of the form $(\forall \xi \leq \overline{\mathsf{n}})\varphi(\xi)$.[11] But this is equivalent to a conjunction $\varphi(\mathsf{0}) \wedge \varphi(\mathsf{1}) \wedge \ldots \wedge \varphi(\overline{\mathsf{n}})$, where each conjunct is a Δ_0 sentence of degree no greater than k, so its truth-value is mechanically calculable; and hence the truth-value of χ is calculable too.

iii. χ is of the form $(\exists \xi \leq \overline{\mathsf{n}})\varphi(\xi)$. The argument is similar.

In sum, we can mechanically settle the truth-value of Δ_0 sentences of degree 0; also, if we can settle the truth-values of Δ_0 sentences of degree up to k, we can settle the truth-values of sentences of degree up to $k+1$. Therefore, by an informal induction on k, we can mechanically settle the truth-values of *all* Δ_0 sentences, however complex they are. ⊠

7.7 Q is Σ_1-complete

Two last definitions in this chapter. Suppose Γ is some class of wffs, and T is an interpreted theory. Then we say

i. T is Γ-*sound* iff, for any sentence $\varphi \in \Gamma$, if $T \vdash \varphi$, then φ is true.

ii. T is Γ-*complete* iff, for any sentence $\varphi \in \Gamma$, if φ is true, then $T \vdash \varphi$.

(Here, 'true' of course means true on the standard interpretation built into T.) And with that jargon to hand, we can state the following theorem, the key result of this chapter:

Theorem 9 Q *is* Σ_1*-complete.*

We can demonstrate this by proving in turn that

1. Q correctly decides every atomic Δ_0 sentence.[12]

2. Q correctly decides every Δ_0 sentence.

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

[12] 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, $\mathsf{Q} \vdash \neg\varphi$.

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 free 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 $\tau_1 = \overline{\mathsf{t}}_1$ and $\tau_2 = \overline{\mathsf{t}}_2$ with numerals on the right. But since '$\leq$' captures the less-than-or-equal-to relation, Q correctly decides whether $\overline{\mathsf{t}}_1 \leq \overline{\mathsf{t}}_2$. Hence, plugging in the identities, Q correctly decides whether $\tau_1 \leq \tau_2$. ⊠

Proof for (2) Again, we 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 again 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 $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ and $(\varphi \leftrightarrow \psi)$. And so Q correctly decides χ.

ii. χ is of the form $(\forall \xi \leq \overline{\mathsf{n}})\varphi(\xi)$. If χ is a *true* sentence, then $\varphi(\mathsf{0})$, $\varphi(\mathsf{1})$, …, $\varphi(\overline{\mathsf{n}})$ must all be true sentences. Being of lower degree, these are – by hypothesis – all correctly decided by Q; so Q proves $\varphi(\mathsf{0})$, $\varphi(\mathsf{1})$, …, $\varphi(\overline{\mathsf{n}})$. Hence, by Result 5 of Section 7.4, we can conclude $(\forall \xi \leq \overline{\mathsf{n}})\varphi(\xi)$. On the other hand, if χ is *false*, $\varphi(\overline{\mathsf{k}})$ is false for some $k \leq n$, and – being of lower degree – this is correctly decided by Q, so Q proves $\neg\varphi(\overline{\mathsf{k}})$. Hence, by Result 6, Q proves $(\exists \xi \leq \overline{\mathsf{n}})\neg\varphi(\xi)$, which easily gives us $\neg(\forall \xi \leq \overline{\mathsf{n}})\varphi(\xi)$. So Q correctly decides χ, i.e. $(\forall \xi \leq \overline{\mathsf{n}})\varphi(\xi)$.

iii. χ is of the form $(\exists \xi \leq \overline{\mathsf{n}})\varphi(\xi)$. 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 $\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ must be true. Therefore, by (2), Q proves $\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ and

hence $\exists x \exists y \varphi(x, y)$, 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. ⊠

7.8 Intriguing corollaries

This chapter has had an amount of tiresome detail. So let's finish on a high note, with a couple of intriguing corollaries of our principal theorem that Q is Σ_1-complete.

We start with a simple observation: 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.[13] As we've noted before, no proof of the conjecture is currently known.

Now suppose that Goldbach's conjecture's is *false*. Then its *negation* will be a true Σ_1 sentence, and hence – by Theorem 9 – be provable in Q. While if Goldbach's conjecture's is *true*, then its negation will be false and so not provable (since Q's axioms are true, so the theory only implies truths). So to determine whether Goldbach's conjecture is true, it is enough to decide whether its negation follows logically from the axioms of Q. Similarly for any Π_1 arithmetical claim (Fermat's last theorem is another example):

> **Theorem 10** *A Π_1 sentence is true if and only if its negation is not logically deducible in* Q *(i.e. if and only if it is consistent with* Q*).*

So, if we had a method which we could use to decide whether a wff was logically deducible from given assumptions, we'd have a method for deciding the truth or falsity of any Π_1 statement of arithmetic. Unfortunately, as we'll later show, there can be no such method (matters of deducibility in first-order logic are not effectively decidable: that's Theorem **??**).

Here's a further development. Let's say a theory T extends Q if it can prove all Q's theorems and retains the same interpretation for those theorems.[14] Since Q is Σ_1-complete, so is any theory T which extends Q.

It immediately follows that

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

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

And the property of being prime can be expressed by the Δ_0 wff

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

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

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

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

[14] 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.

Theorem 11 *If T extends* Q*, T is consistent iff it is Π_1-sound.*

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

The converse is trivial, since if T is inconsistent, we can derive anything in T, including false Π_1 sentences 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. Suppose that some really ingenious deduction of the Goldbach conjecture 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).

8 First-order Peano Arithmetic

We put some effort into discussing what Q can prove because 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 an extremely weak theory. To derive elementary general truths like $\forall x(0 + x = x)$ that are beyond Q's reach, we evidently need a formal arithmetic that incorporates some stronger axiom(s) for proving quantified wffs. This chapter develops the natural *induction axioms* we need to add, working up to the key theory PA – 'first-order Peano 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. Take just one example. To prove that Q correctly decides all Σ_1 wffs we in effect said: let n have the property P if Q correctly decides all Σ_1 wffs of degree no more than n. 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 correctly decides any Σ_1 wff, whatever its degree.

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

[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 0 has property P and the supposition that *every* number less than n has P implies that n has P, then every number has property P. And there is 'the method of infinite descent', which is the principle that if the supposition that n has P' always implies there is a smaller number $m < n$ such that m has P', then 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.

intuitive induction principle is therefore underwritten by the basic structure of 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 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 $(\{\varphi(0) \wedge \forall x(\varphi(x) \rightarrow \varphi(Sx))\} \rightarrow \forall x\varphi(x))$

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

(c) We'll explain what counts as 'suitable' in a moment. But even before we spell that out, we can see that we'll need to generalize the story a bit. That's 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 some arithmetical 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):

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

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

$$\forall y\forall z(\{R0yz \wedge \forall x(Rxyz \rightarrow R(Sx)yz)\} \rightarrow \forall xRxyz)$$

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 $\varphi(x) = Rxyz$ into our original Induction Schema: i.e. it's what you

[2] Compare BA where we had to use schemata 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*.

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.

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 is to count as an axiom, where* $\varphi(\mathsf{x})$ *in the Schema now holds the place of a suitable open wff that may also have variables other than 'x' free.*[3]

8.2 Arguing using Δ_0 induction

But which wffs φ are 'suitable' for appearing in the induction schema? Well, let's start very modestly. Take a wff $\varphi(\mathsf{x})$ which is bounded, i.e. is Δ_0. Then such a wff surely expresses an entirely determinate property: indeed, for any n, we can quite mechanically decide whether n has that property, i.e. we can decide whether $\varphi(\overline{\mathsf{n}})$ is true or not – see Section 7.5 (c). Similarly, a Δ_0 wff with two or more free variables surely expresses an entirely determinate relation. So in this section, let's consider the quite uncontroversial use of induction for Δ_0 wffs φ.

Let's use the standard label $\mathsf{I}\Delta_0$ for the theory you get by adding to Q the (universal closure of) instances of the Induction Schema where φ is Δ_0. Then the following, for example, will all be theorems of $\mathsf{I}\Delta_0$:

1. $\forall \mathsf{x}(0 + \mathsf{x} = \mathsf{x})$
2. $\forall \mathsf{x}\forall \mathsf{y}(\mathsf{Sx} + \mathsf{y} = \mathsf{S}(\mathsf{x} + \mathsf{y}))$
3. $\forall \mathsf{x}\forall \mathsf{y}(\mathsf{x} + \mathsf{y} = \mathsf{y} + \mathsf{x})$
4. $\forall \mathsf{x}\forall \mathsf{y}(\mathsf{x} \leq \mathsf{y} \vee \mathsf{y} \leq \mathsf{x})$.
5. $\forall \mathsf{x}(\mathsf{x} \neq \mathsf{Sx})$

We had better sketch how to prove such wffs inside $\mathsf{I}\Delta_0$. However, these are again 'trivial but tiresome' results. So don't get bogged down: by all means skim and or just skip to the next section.

Proof for (1) We showed in Section 6.4 that the wff $\forall \mathsf{x}(0 + \mathsf{x} = \mathsf{x})$ is *not* provable in Q. But we can derive it by a Δ_0 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 $\forall \mathsf{x}(0 + \mathsf{x} = \mathsf{x})$, we will need to start with an instance of the Induction

[3] If we allow axioms to be wffs with free variables, then – in the presence of the usual logical rules – an equivalent alternative is to allow *any* instance of the Induction Schema to be an axiom, whether or not it has variables dangling free.

Schema with $\varphi(\mathsf{x})$ replaced by the Δ_0 wff $(0 + \mathsf{x} = \mathsf{x})$. 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, for what it is worth, is a full-dress formal version:

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

So, laboriously, we are done. ⊠

Proof for (2) We'll derive the equivalent $\forall \mathsf{y} \forall \mathsf{x}(\mathsf{Sy} + \mathsf{x} = \mathsf{S}(\mathsf{y} + \mathsf{x}))$. And since full-dress proofs rapidly become tedious, we'll just describe the proof in outline. So we start with the universal closure of an instance of the Induction Schema with $\varphi(\mathsf{x})$ replaced by $(\mathsf{Sy} + \mathsf{x} = \mathsf{S}(\mathsf{y} + \mathsf{x}))$.

$$\forall \mathsf{y}[\{\mathsf{Sy} + 0 = \mathsf{S}(\mathsf{y} + 0) \wedge \forall \mathsf{x}(\mathsf{Sy} + \mathsf{x} = \mathsf{S}(\mathsf{y} + \mathsf{x}) \rightarrow \mathsf{Sy} + \mathsf{Sx} = \mathsf{S}(\mathsf{y} + \mathsf{Sx}))\} \rightarrow \ \forall \mathsf{x}(\mathsf{Sy} + \mathsf{x} = \mathsf{S}(\mathsf{y} + \mathsf{x}))]$$

And, of course, we then instantiate the initial universal quantifier to get

$$\{\mathsf{Sa} + 0 = \mathsf{S}(\mathsf{a} + 0) \ \wedge \ \forall \mathsf{x}(\mathsf{Sa} + \mathsf{x} = \mathsf{S}(\mathsf{a} + \mathsf{x}) \rightarrow \mathsf{Sa} + \mathsf{Sx} = \mathsf{S}(\mathsf{a} + \mathsf{Sx}))\} \rightarrow \ \forall \mathsf{x}(\mathsf{Sa} + \mathsf{x} = \mathsf{S}(\mathsf{a} + \mathsf{x}))$$

But we can derive $\mathsf{Sa} + 0 = \mathsf{S}(\mathsf{a} + 0)$, the first conjunct in the curly brackets, by two uses of Axiom 4. So now we aim for the other conjunct. Still arguing in Q, let's suppose $\mathsf{Sa} + \mathsf{b} = \mathsf{S}(\mathsf{a} + \mathsf{b})$ as a temporary assumption. Using Axiom 5 twice gives us $\mathsf{Sa} + \mathsf{Sb} = \mathsf{S}(\mathsf{Sa} + \mathsf{b}) = \mathsf{SS}(\mathsf{a} + \mathsf{b}) = \mathsf{S}(\mathsf{a} + \mathsf{Sb})$. So using Conditional Proof and then generalizing on b gives us the second conjunct in the curly brackets. Hence we will be able to derive $\forall \mathsf{x}(\mathsf{Sa} + \mathsf{x} = \mathsf{S}(\mathsf{a} + \mathsf{x}))$. Finally we generalize again to get the desired conclusion. ⊠

Proof for (3) Take the universal closure of the instance of induction for $\varphi(\mathsf{x}) = \mathsf{x} + \mathsf{y} = \mathsf{y} + \mathsf{x}$, and then instantiate to get

$$\{0 + \mathsf{a} = \mathsf{a} + 0 \wedge \forall \mathsf{x}(\mathsf{x} + \mathsf{a} = \mathsf{a} + \mathsf{x} \rightarrow \mathsf{Sx} + \mathsf{a} = \mathsf{a} + \mathsf{Sx})\} \rightarrow \ \forall \mathsf{x}(\mathsf{x} + \mathsf{a} = \mathsf{a} + \mathsf{x})$$

To proceed, we can obviously derive $0 + a = a + 0$, the first conjunct in the curly brackets, by using the Result 1 plus Axiom 4. And now we can derive the second conjunct, noting that for arbitrary b, given $b + a = a + b$, we can use Axiom 5 and Result 2 above to derive $b + Sa = S(a + b) = Sa + b$. So by Conditional Proof $b + a = a + b \rightarrow Sb + a = a + Sb$. So we can generalize to get that second conjunct. And then we can detach the consequent and generalize again to get what we wanted. ⊠

Proof for (4) Swapping variables, let's show $\forall y \forall x(y \leq x \vee x \leq y)$. Take the universal closure of the instance of induction for $\varphi(x) = y \leq x \vee x \leq y$, and instantiate to get

$$\{(a \leq 0 \vee 0 \leq a) \wedge \forall x((a \leq x \vee x \leq a) \rightarrow (a \leq Sx \vee Sx \leq a))\} \rightarrow \forall x(a \leq x \vee x \leq a)$$

So again we need to prove each conjunct in the curly brackets; then we can detach the consequent of the conditional, and generalize to get the desired result.

It's trivial to prove the first conjunct, $(a \leq 0 \vee 0 \leq a)$, since its second disjunct always obtains, by Result 1 of Section 7.4.

Now we show that $(a \leq b \vee b \leq a) \rightarrow (a \leq Sb \vee Sb \leq a)$ for arbitrary b, which will prove the second conjunct. Argue by cases.

Suppose first $a \leq b$, i.e. $\exists v(v + a = b)$. But if for some c, $c + a = b$, then $S(c + a) = Sb$, so by Result 2 above, $Sc + a = Sb$, whence $\exists v(v + a = Sb)$, i.e. $a \leq Sb$, so $(a \leq Sb \vee Sb \leq a)$.

Suppose secondly $b \leq a$, i.e. $\exists v(v + b = a)$. Then for some c, $c + b = a$. By Q's Axiom 3, either $c = 0$, or $c = Sc^\circ$ for some c°. In the first case $0 + b = a$, so by Result 1 above $a = b$, from which it is trivial that $a \leq Sb$. In the second case, $Sc^\circ + b = a$, and using Result 2 and Axiom 5 we get $c^\circ + Sb = a$, and hence $Sb \leq a$. So $(a \leq Sb \vee Sb \leq a)$ again. Phew! ⊠

Which all goes to show that proofs by induction do indeed soon get *very* tiresome. But carrying on in the same way, we can in fact prove a lot of the most obvious general properties of addition and multiplication even in $I\Delta_0$.

We end with a trivially easy case. Recall that our deviant interpretation which makes the axioms of Q true while making $\forall x(0 + x = x)$ false has Gwyneth as a self-successor. Induction, however, rules out self-successors:[4]

Proof for (5) Take φ to be $(x \neq Sx)$ (another Δ_0 wff). Then Q already entails $\varphi(0)$ (that's Axiom 1), and entails $\forall x(\varphi(x) \rightarrow \varphi(Sx))$ (by contraposing Axiom 2). So by induction, $\forall x \varphi(x)$, i.e. no number is a self-successor. ⊠

[4]Don't be lulled into a false sense of security, though! While $I\Sigma_1$'s axioms may rule out deviant interpretations based on self-successors, they don't rule out some other deviant interpretations.

8.3 PA – First-order Peano Arithmetic

We said: (the universal closure of) any instance of the Induction Schema will be intuitively acceptable as an axiom, so long as we replace φ in the schema by a suitable open wff. Now to repeat the question that we posed before: which open wffs of L_A are 'suitable'?

We argued at the beginning of Section 8.2 that Δ_0 wffs are eminently suitable, and we have so far used only examples of induction involving such wffs. And it is certainly of technical interest to see how many basic truths of arithmetic can in fact be proved by adding just that limited amount of induction to Q. But why be so restrictive?

Take *any* open wff φ of L_A. When any abbreviatory symbols are unpacked, this will be constructed from no more than the constant term '$\mathsf{0}$', the successor, addition and multiplication functions, plus identity and other logical apparatus: therefore – you might very well suppose – it ought also to express a perfectly determinate arithmetical property or relation (even if, in the general case, we can't always decide whether a given number n has the property or not). So why not be generous and allow *any* open wff of L_A to be substituted for φ in the Schema?

Here's a positive argument for generosity. Remember that instances of the Induction Schema are (quantified) *conditionals*. So, for example, they actually only allow us to derive $\forall \mathsf{x}\varphi(\mathsf{x})$ when we can *already* prove (i) $\varphi(\mathsf{0})$ and also prove (ii) $\forall \mathsf{x}(\varphi(\mathsf{x}) \rightarrow \varphi(\mathsf{Sx}))$. But if we can already prove (i) and (ii) then (iii) we can already prove each and every one of $\varphi(\mathsf{0})$, $\varphi(\mathsf{S0})$, $\varphi(\mathsf{SS0})$ .... But if we *can* already prove that each $\varphi(\overline{\mathsf{n}})$ is true, then what more can it take for φ to express a genuine property that indeed holds for every number? While on the other hand, if we *can't* already prove each $\varphi(\overline{\mathsf{n}})$, i.e. can't prove (i) and (ii), then accepting the instance of the Induction Schema for φ must be harmless. It then just doesn't matter whether we think φ expresses a kosher property or not: we won't be able to use that instance of the Induction Schema for φ to prove anything, because we won't be able to detach the consequent of the conditional.

It seems, therefore that we can't overshoot by allowing (the universal closures of) instances of the Induction Schema for *any* open wff φ. The only *usable* instances of our generous helping of axioms will be the axioms which we will actually have perfectly good reason to accept.

So let's now take it that any open wff of L_A can be used in the Induction Schema. That means moving on from Q and $\mathsf{I}\Delta_0$ – and jumping right over a range of possible intermediate theories[5] – to adopt the much richer formal theory

[5]Later, we'll have reason to highlight the highlight the theory $\mathsf{I}\Sigma_1$, which is what you get when you add to Q all instances of induction for Σ_1 wffs. For an extended exploration of the powers of this and other intermediate theories of arithmetic with various restrictions on the Induction Schema, see the wonderful (Hájek and Pudlák, 1993). These theories are indeed metalogically very interesting: but, to repeat, the *natural* theory of arithmetic-with-induction is the theory we are jumping to, namely PA.

of arithmetic that we can briskly define as follows:

> PA – First-order Peano Arithmetic[6] – is the result of adding to the axioms of Q the universal closures of *all* instances of the Induction Schema.

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

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. 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. Evidently, PA can prove those results which we showed in the previous section can be demonstrated using induction just for Δ_0 formulae. But PA now also allows induction for more complex formulae, including e.g. the Σ_1 formula

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

But $\varphi(\mathsf{0})$ is a trivial PA theorem. Likewise, $\forall \mathsf{x}\varphi(\mathsf{Sx})$ is also a trivial theorem, and that entails $\forall \mathsf{x}(\varphi(\mathsf{x}) \rightarrow \varphi(\mathsf{Sx}))$. So we can use an instance of the Induction Schema inside PA to derive $\forall \mathsf{x}\varphi(\mathsf{x})$. But that's just Axiom 3 of Q.[7] So our initial 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 ...

8.4 Summary overview of PA

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 official deductive *proof system* is a Hilbert-style axiomatic version of classical first-order logic with identity (the differences between various presentations of first-order logic of course don't make a difference to what can be proved in PA: our choice is just for later metalogical convenience: see Sections 7.1 and **??**).

[6] 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).

[7] As we saw in Section 7.4, 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.

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

Axiom 1 $\forall x(0 \neq Sx)$

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

Axiom 3 $\forall x(x + 0 = x)$

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

Axiom 5 $\forall x(x \times 0 = 0)$

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

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

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

where $\varphi(x)$ is an open wff that has 'x', and perhaps other variables, free.

8.5 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 this 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.[8] Note, though, that the availability of a complete theory for successor and addition was proved as early as 1929 by Mojzesz Presburger.[9]

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

[8] 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).

[9] To be strictly accurate, Presburger – then a young graduate student – proved the completeness not of P but of a rather different and less elegant theory (whose primitive expressions were '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'.

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 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. It wasn't obviously predictable that multiplication would make all the difference.[10]

8.6 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 entailing *every* wff. So let's finish the main business of 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). And that informal induction principle, warrants the usable instances of PA's Induction Schema: so they too 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 *both* to talk about wffs of PA *and* to be able to describe the interpretation $\mathcal{I}_A$. 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 knows enough about the workings of $\mathcal{I}_A$, it will prove e.g. a formal wff that says '$\forall$x(0 $\neq$ Sx)' is true iff $\forall x(0 \neq Sx)$. But to go on to prove that PA's first

[10] 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.

Axiom actually *is* true, T will then itself need to prove $\forall x(0 \neq Sx)$. In other words, T must incorporate something equivalent to PA's first axiom. 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.

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*.[11]

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,[12] 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.[13] 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;

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

[12] 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).

[13] See (Russell, 1902).

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 supposed 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: more on this in Chapter **??**.

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? Even 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:

$$0! = S0 = 1$$
$$(Sy)! = y! \times Sy$$

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:

$$x^0 = S0$$
$$x^{Sy} = (x^y \times x)$$

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:

$$x \times 0 = 0$$
$$x \times Sy = (x \times y) + x$$

$$x + 0 = x$$
$$x + Sy = S(x + y)$$

Three comments about our examples so far:

i. In each definition, the second clause fixes the value of a function for argument Sn by invoking the value of the *same* function for argument n. This kind of procedure is standardly termed 'recursive'. And our two-clause definitions are examples of *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, but 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:

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

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

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

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 that's 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 here 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:

$$f(x, 0) = g(x)$$
$$f(x, Sy) = h(x, y, f(x, y))$$

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, $I^k_i(x_1, x_2, \ldots, x_k) = x_i$ 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 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 is by definition one that *can* be specified by a certain sort of chain of definitions. And so the natural way of presenting such a function 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

> $fermat(n) = n$ if there are solutions to $x^{n+3} + y^{n+3} = z^{n+3}$ (with x, y, z positive integers);
> $fermat(n) = 0$ 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 indeed have to treat properties and functions in the same way. For Frege urges us to regard 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.

The salient point is this: 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) it is trivial that the initial functions S, Z, and I_i^k are algorithmically 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 observation – if g and h are algorithmically 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 algorithmically computable functions.

To illustrate (c), return once more to our example of the factorial. Here is 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. $\quad fact := (fact \times Sy)$
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 $n!$.

More generally, for any 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

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

with the corresponding program

1. $func := g$
2. For $y = 0$ to $n - 1$
3. $\quad func := h(y, func)$
4. 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

1. $func := g(m)$
2. For $y = 0$ to $n - 1$
3. $\quad func := h(m, y, func)$
4. 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

$$\text{factorial} \Rightarrow \text{multiplication} \Rightarrow \text{addition} \Rightarrow \text{successor}$$

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 algorithmically 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 algorithmically 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

[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.

particular, we allowed computations to involve open-ended searches, with no 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

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

generate 'in alphabetical order' all possible strings of symbols from this language; and as we go along, 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.

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 effectively computable, using a step-by-step algorithmic procedure.

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 an algorithmically computable function which is not primitive recursive. ⊠

'But hold on! *Why* is the diagonal function not a p.r. function?' Well, consider 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 successive computations of all the different functions f_n which enables them to be wrapped up into a single p.r. function. And our diagonal argument in effect shows that this can't be done.

9.6 Defining p.r. properties and relations

The *p.r. functions* 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 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 *effectively decidable* – i.e. a suitably programmed computer can decide whether the property obtains – just if its *characteristic function* is *(total and) effectively 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.

So, given that any p.r. function is effectively computable, p.r. properties are among the effectively decidable ones.

9.7 Building p.r. functions and relations

The last two sections of this chapter give some general principles for building new p.r. functions and relations out of old ones, and then give some examples of some of these principles at work.

Strictly speaking, these are more 'trivial but tiresome' details which you can cheerfully skip, since we only pick up their details in other sections that you can also skip. On the other hand, in proving Gödel's Theorems, we will need to claim that a variety of 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 these sections: but if you have no taste for this sort of detail, don't worry.

A couple of definitions before we get down to business. First, 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

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

[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.

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.

Second, suppose that the function f is defined in terms of k other p.r. functions f_i as follows

$$\begin{array}{l} f(n) = f_0(n) \text{ if } C_0(n) \\ f(n) = f_1(n) \text{ if } C_1(n) \\ \quad \vdots \\ f(n) = f_k(n) \text{ if } C_k(n) \\ f(n) = a \text{ otherwise} \end{array}$$

where the conditions C_i are exclusive and express p.r. properties (i.e. have p.r. characteristic functions c_i), and a is a constant. Then f is said to be *defined by cases* from other p.r. functions

Now we can state five useful general facts about function/relation building:

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.

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.

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.

E. Any function defined by cases from other p.r. functions is also p.r.

These claims (A) to (E) should all look intuitively plausible (why?). But we'd better give official proofs.

A preliminary result Put $sg(n) = 0$ for $n = 0$, and $sg(n) = 1$ otherwise. Then sg is primitive recursive. For we just note that

$$\begin{array}{l} sg(0) = 0 \\ sg(Sy) = SZ(sg(y)) \end{array}$$

where $SZ(u)$ is p.r. by composition, and $SZ(sg(y)) = S0 = 1$. Also, let $\overline{sg}(n) = 1$ for $n = 0$, and $\overline{sg}(n) = 0$ otherwise. Then $\overline{sg}$ is similarly shown to be p.r. ⊠

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 $\overline{sg}(p(x))$ is the characteristic function of the property *not-P*, since $\overline{sg}$ simply flips the two values 0 and 1. But by simple composition of functions, $\overline{sg}(p(x))$ 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

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

More carefully, suppose that $p(x)$ is P's p.r. characteristic function. And by composition define the p.r. function $h(u, v) = (p(Su) \times v)$. 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 in the bounded quantifiers we could 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. ⊠

Proof for (E) Just note that

$$f(n) = \overline{sg}(c_0(n))f_0(n) + \overline{sg}(c_1(n))f_1(n) + \ldots + \overline{sg}(c_k(n))f_k(n) + c_0(n)c_1(n)\ldots c_k(n)a$$

since $\overline{sg}(c_i(n)) = 1$ when $C_i(n)$ and is otherwise zero, and the product of all the $c_i(n)$ is 1 just in case none of $C_i(n)$ are true, and is zero otherwise. ⊠

9.8 Some more examples

(a) With our shiny new building tools to hand, we can finish the chapter by giving a few more examples of p.r. functions, properties and relations, and then proving that they *are* p.r.

1. The relations $m = n$, $m < n$ and $m \leq n$ are primitive recursive.
2. The relation $m|n$ that holds when m is a factor of n is primitive recursive.
3. Let $Prime(n)$ be true just when n is a prime number. Then $Prime$ is a p.r. property.[9]
4. List the primes as $\pi_0, \pi_1, \pi_2, \ldots$. Then the function $\pi(n)$ whose value is π_n is p.r.
5. 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.
6. 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.

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

(b) We'll now show that those examples are, as claimed, all primitive recursive. However – like the arguments in Section 7.4 – the mini-proofs that follow don't involve anything deep or illuminating. And do remember, 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) 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 primitive recursive. ⊠

Proof for (2) 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 of the last section). So that relation is p.r. by Fact B. Hence $m|n$ is a p.r. relation by Fact C. ⊠

Proof for (3) The property of being *Prime* is p.r. because

$$Prime(n) \leftrightarrow n \neq 1 \ \wedge \ (\forall u \leq n)(\forall v \leq n)(u \times v = n \rightarrow \ (u = 1 \vee v = 1))$$

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

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

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 (5) By the Fundamental Theorem of Arithmetic, which says that numbers have a unique factorization into primes, this function is well-defined. And no exponent in the prime factorization of n is larger than n itself, so

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

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 $exp(n, k) = 0$ 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 (6) $(Prime(m) \wedge m|n)$ 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 $pf(m, n)$. Now consider the function

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

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

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

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

$$l(x, 0) = p(0, x)$$
$$l(x, Sy) = (p(Sy, x) + l(x, y))$$

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

Well, that's enough to be going on with. And 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 tiresome details of how to do it.

10 Capturing functions

In this chapter we introduce the pivotal idea of a *p.r. adequate theory* of arithmetic, i.e. one that can appropriately 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 the idea of capturing a *function* as opposed to a property or relation. And there's a snag. In fact we find *three* different ideas at different places in the literature. I'll officially call them *capturing*, *capturing as a function*, and *fully defining a function* respectively (though just that's my jargon: predictably, terminology is a complete mess here, and e.g. 'defines' has been variously used for each of them). The first couple of sections do the necessary housekeeping, sorting things out.

10.1 Three ways of capturing a function

(a) 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: f and R_f have 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 (total functions, that is: I won't keep repeating the point that, throughout this chapter, all the functions we are considering are functions defined for every numerical input value).

For a start, then, 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 $\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is true,
>
> if m doesn't have relation R_f to n, then $\neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is true.

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

[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.

> 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 $\varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$ is true,
> if $f(m) \neq n$, then $\neg\varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$ is true.

The generalization to many-place functions is immediate.

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 $T \vdash \varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$,
> if m does not have the relation R_f to n, then $T \vdash \neg\varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$.

And we can naturally say that a one-place total function f 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 this, our first key definition:

A. 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 $T \vdash \varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$,
if $f(m) \neq n$, then $T \vdash \neg\varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$.

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

(b) 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 very common to work with a slightly stronger notion of capturing a function.

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:

B. 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, $T \vdash \exists!\mathsf{y}\varphi(\overline{\mathsf{m}},\mathsf{y})$
and for any m, n:
(ii) if $f(m) = n$ then $T \vdash \varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$,
(iii) if $f(m) \neq n$, then $T \vdash \neg\varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$.

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

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

capturing relation can be proved to relate each numerical argument to some unique value: in a phrase, the relation is (provably) functional for numbers.

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

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 $T \vdash \varphi(\overline{\mathsf{m}}, \overline{\mathsf{k}})$. Then by simple logic, (i) and (ii) imply that $T \vdash \overline{\mathsf{n}} \neq \overline{\mathsf{k}} \rightarrow \neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$. But as we saw in Section 6.1, even when T is mere Baby Arithmetic, if $n \neq k$, then $T \vdash \overline{\mathsf{n}} \neq \overline{\mathsf{k}}$. Hence, if T contains Baby Arithmetic, (iii) if $f(m) \neq n$ then $T \vdash \neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$. ⊠

Therefore, to confirm 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 another common definition of basically the same idea of capturing-as-a-function, 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 (for total functions, remember):

> 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 $T \vdash \forall\mathsf{y}(\varphi(\overline{\mathsf{m}}, \mathsf{y}) \leftrightarrow \mathsf{y} = \overline{\mathsf{n}})$.

Likewise, of course, for many-place functions.

(c) We remarked at the beginning of the previous chapter that arithmetics like Q and PA only have just three functions built in. In the next chapter, however, we are going to prove that Q (and hence PA) can in fact capture any p.r. function – and indeed, can capture it as a function. Still, this sort of capturing as we've just defined it is done by *relational* expressions: what about capturing functions with new *functional* expressions? (After all, it is much more natural to represent functions by function symbols!)

Suppose, then, that the L_A wff $\varphi(\mathsf{x}, \mathsf{y})$ captures the function $f(x) = y$ in theory T. It would be convenient, for ease of notation, to expand the language L_A by adding a corresponding function symbol 'f', and expand T by adding as a new axiom the following principle to define it:

$$\forall\mathsf{x}\forall\mathsf{y}\{\mathsf{f}(\mathsf{x}) = \mathsf{y} \leftrightarrow \varphi(\mathsf{x}, \mathsf{y})\}$$

However, note that extending T by what is intended to be a mere notational convenience shouldn't make any difference to which wffs of the original, unextended, language are provable: in a probably familiar jargon, this extension of T should be a *conservative* one. And the condition for it to be conservative to add the function symbol 'f' with its definition is that T can prove $\forall\mathsf{x}\exists!\mathsf{y}\varphi(\mathsf{x}, \mathsf{y})$. That's an entirely standard result about first-order theories and we won't prove it here.[3]

[3] See e.g. (Enderton, 2002, Theorem 27A, p. 165) or (Mendelson, 1997, pp. 103–104).

But this *is* a stronger requirement than is given in our idea of φ's capturing f as a function – for a weak theory like Q might be able to prove $\exists!\mathsf{y}\varphi(\overline{\mathsf{m}}, \mathsf{y})$ for each number m and not be able to prove the generalization $\forall\mathsf{x}\exists!\mathsf{y}\varphi(\mathsf{x}, \mathsf{y})$.

One more definition then! Let's say that

C. The one-place function f is *fully defined as a function* by $\varphi(\mathsf{x}, \mathsf{y})$ in theory T just if
   (i) $T \vdash \forall\mathsf{x}\exists!\mathsf{y}\varphi(\mathsf{x}, \mathsf{y})$
   (ii) if $f(m) = n$ then $T \vdash \varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$

Likewise for many place functions, of course. Then, if a function can be fully defined as a function by some relational expression of T, then we can legimitately introduce a function symbol for f into T via a definitional principle of the kind we gave. (And if T *already* contains a suitable function symbol, then $\mathsf{f}(\mathsf{x}) = \mathsf{y}$ will do the job of fully defining as a function.)[4]

10.2 Relating our definitions

(a) Trivially, if (C) φ fully defines f as a function in T, then (B) φ captures f as a function. And if (B) φ captures f as a function in T, then (A) φ captures f in the weak sense. The converse implications don't strictly obtain. However, we have results which are almost as good.

Suppose T is either Q or extends Q, so T proves everything that Q does: then, firstly, if (A) φ captures the function f in T, then (B′) there will always be a closely related wff $\widetilde{\varphi}$ which *does* capture f *as a function* in T.

Let's illustrate this 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, $\widetilde{\varphi}(\overline{\mathsf{m}}, \mathsf{x})$ is satisfied by a *unique* n, i.e. the smallest n such that $\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ 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 7.4, Q is good at proving results involving bounded quantifiers. In tiresome detail (and this is 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

[4] We have laboured over these variant A/B/C definitions in part to help comparison with the ideas in other books. As we said at the beginning of the chapter, 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).

Finally, when e.g. (Hájek and Pudlák, 1993, p. 47) or (Buss, 1998, p. 87) talk of a formula defining a function *they* mean what we are calling fully defining a function.

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

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) $T \vdash \varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$, and (b) for $k < n$, $T \vdash \neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{k}})$. But (a) and (b) imply (c): for $k \leq n$, $T \vdash \varphi(\overline{\mathsf{m}}, \overline{\mathsf{k}}) \rightarrow \overline{\mathsf{k}} = \overline{\mathsf{n}}$. And (c) and Result 5 of Section 7.4 entail (d) $T \vdash (\forall \mathsf{x} \leq \overline{\mathsf{n}})(\varphi(\overline{\mathsf{m}}, \mathsf{x}) \rightarrow \mathsf{x} = \overline{\mathsf{n}})$. Putting (a) and (d) together, that means $T \vdash \widetilde{\varphi}(\overline{\mathsf{m}}, \overline{\mathsf{n}})$, which establishes (ii).

Since $T \vdash \widetilde{\varphi}(\overline{\mathsf{m}}, \overline{\mathsf{n}})$, to establish (i) it is now enough to show that, for arbitrary a, $T \vdash \widetilde{\varphi}(\overline{\mathsf{m}}, \mathsf{a}) \rightarrow \mathsf{a} = \overline{\mathsf{n}}$. So, arguing in T, suppose $\widetilde{\varphi}(\overline{\mathsf{m}}, \mathsf{a})$, i.e. $\varphi(\overline{\mathsf{m}}, \mathsf{a}) \wedge (\forall \mathsf{z} \leq \mathsf{a})(\varphi(\overline{\mathsf{m}}, \mathsf{z}) \rightarrow \mathsf{z} = \mathsf{a})$. By Result 9 of Section 7.4, $\mathsf{a} \leq \overline{\mathsf{n}} \vee \overline{\mathsf{n}} \leq \mathsf{a}$. If the first, (d) yields $\varphi(\overline{\mathsf{m}}, \mathsf{a}) \rightarrow \mathsf{a} = \overline{\mathsf{n}}$, and so $\mathsf{a} = \overline{\mathsf{n}}$. If the second, then $\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}}) \rightarrow \overline{\mathsf{n}} = \mathsf{a}$, so $\overline{\mathsf{n}} = \mathsf{a}$. So either way $\mathsf{a} = \overline{\mathsf{n}}$. 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.

We also have the following stronger result. If T includes Q *and* has induction at least for Δ_0 wffs, then if (A) φ captures f in T, then (C′) $\widetilde{\varphi}$ will fully define f as a function. We can leave checking this further claim as an exercise. (Hint: Result 4 of Section 8.2 tells us that Δ_0 induction is enough to prove that, for arbitrary a and b, $\mathsf{a} \leq \mathsf{b} \vee \mathsf{b} \leq \mathsf{a}$. That's what we need to run a generalized version of the proof that (A) implies (B′).)

(b) Let's pause to note one way in which our various definitions, old and new, hang together. Suppose T contains quantifiers, and (like Q) can prove that $\mathsf{0} \neq \mathsf{1}$; 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

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

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 $\varphi(\mathsf{x}, \mathsf{0})$ captures the corresponding property P. ⊠

10.3 The idea of p.r. adequacy

(a) Even BA has function symbols for successor, addition and multiplication built in. But plainly it 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.[5]

However, in Q and PA, successor, addition and multiplication *are* fully defined as functions. To take just one example, $\mathsf{x} + \mathsf{y} = \mathsf{z}$ fully defines addition as a function, as is easily confirmed.

(b) But 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.[6] That motivates a trio of definitions:

> A theory T is *weakly p.r. adequate*/*p.r. adequate*/*strongly adequate* if, for every p.r. function f, there is a corresponding φ in T that captures it/captures it as a function/fully defines 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 schemata which reflect the p.r. definitions of addition and multiplication. As we showed, we can use instances of these schemata 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 schemata. As a first step, we can add the symbol '↑', intended to express the exponential function, and then say that all numeral instances of the following are axioms too:

Schema 7 $\quad \zeta \uparrow 0 = 1$

Schema 8 $\quad \zeta \uparrow \mathsf{S}\xi = (\zeta \uparrow \xi) \times \zeta$

[5] Here we take up the permission we gave ourselves in Section 4.4, fn. 5 to read the variables in the official definitions of expressing/capturing as serving as placeholders when necessary.

[6] 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.

Instances of those schemata 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 'f_0', 'f_1', 'f_2', 'f_3'. And now let's keep going: we will add a symbol 'f_n' for each n, with the plan that '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 mentioned earlier in the enumeration (particular functions will, of course, covered repeatedly in the enumeration of recipes, since any p.r. function can be defined in many ways). Then for each 'f_n', we write down schemata involving that function expression which reflect f_n's recipe defining the function in terms of earlier ones. Call the resulting theory PRA_0.

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 schemata. 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 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, PRA_0 can capture all p.r. functions[7] – though, lacking quantifiers, it of course can't be p.r. adequate in the stronger senses.[8]

(d) In sum, we can readily construct a (weakly) p.r. adequate arithmetic by the rather high-cost method of infinitely expanding the vocabulary of arithmetic and then 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.

[7] Footnote 5 above applies again!

[8] 'PRA_0' is short for 'quantifier-free Primitive Recursive Arithmetic'. For the record, full PRA is PRA_0 with all first-order logic restored *and* with induction for all open wffs that don't involve unbounded quantification. Full PRA therefore has a very rich language but just by construction a p.r. adequate theory in the strongest sense. And in fact some treatments take this theory rather than Q to be their 'core' arithmetical theory just because (i) we don't have the bother of having to *prove* p.r. adequacy, and (ii) it is so much easier to work with a theory where all the p.r. functions are captured using function symbols: see e.g. Smoryński (1985). By contrast, focusing on Q as we do, we have the hard work of proving its p.r. adequacy. But the pay-off is that our later proof of the First Incompleteness Theorem applies even to theories built in the modest language L_A.

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, even 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' in the sense of Section 5.1.

This chapter contains the first heavy-weight proofs in this book, so is unavoidably rather tougher going. Here's a road-map of the overall line of argument. Recall the idea of a Σ_1 wff which was introduced in Section 7.5. A Σ_1 wff is the existential quantification of a Δ_0 kernel wff (where being 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 second section below, building on our work in Section 7.7, which showed us that Q correctly decides every Δ_0 wff and is Σ_1-complete.

2. *Every p.r. function is a* Σ_1 *function.* This takes us another 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, so p.r. functions are Σ_1.

Those two Big Results together immediately entail that Q is p.r. adequate. It trivially follows that PA is (strongly) p.r. adequate too.

Perhaps we should note, though, that the new proof ideas which we need in order to establish the two Big Results are not used again in this book. It is therefore not really necessary to master all the details of the proofs in order to grasp what follows later. Feel free to skim, though do at least read the last couple of sections of the chapter where we sum up.

11.1 More definitions

We start with a trio of simple definitions – definitions which 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.

It is very important to 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 $\forall z(f(m) = z \rightarrow z = n)$. Hence $f(m) = n$ if and only if $\forall \mathsf{z}(\varphi(\overline{\mathsf{m}}, \mathsf{z}) \rightarrow \mathsf{z} = \overline{\mathsf{n}})$ is true.[1] In other words, f is equally well expressed by $\forall \mathsf{z}(\varphi(\mathsf{x}, \mathsf{z}) \rightarrow \mathsf{z} = \mathsf{y})$. But it is a trivial exercise of moving quantifiers around to show that if $\varphi(\mathsf{x}, \mathsf{y})$ is strictly Σ_1, then $\forall \mathsf{z}(\varphi(\mathsf{x}, \mathsf{z}) \rightarrow \mathsf{z} = \mathsf{y})$ is Π_1. ⊠

11.2 Q can capture all Σ_1 functions

(a) In Section 7.7, 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 that if Q captures a function by some wff φ, it can capture it as a function by a corresponding wff $\widetilde{\varphi}$ (for details, see Section 10.2 again). These results immediately entail

Theorem 13 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 $\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is true, and hence – since Q correctly settles every Δ_0 wff, $\mathsf{Q} \vdash \varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$. Likewise, if $f(m) \neq n$, then $\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is false, and hence $\mathsf{Q} \vdash \neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$. 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. 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.) ⊠

(b) The rest of this section beefs up that last very easy theorem by using a delightful bit of sheer ingenuity to establish the stronger result

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 a related wff which *captures* f as a function: and in fact it will still be a Σ_1 wff (and could be the same one).

Proof Our proof falls into two stages. First (the really ingenious stage), we show that a Σ_1 function is equivalent to a composition of Δ_0 functions. And then

[1] To avoid clash of variables, assume 'z' doesn't appear in φ.

second (the straightforward stage), we show that Q can capture any composition of Δ_0 functions.

First stage Take a total one-place function $f(x) = y$ expressed by a strictly Σ_1 open wff with two free variables. Suppose first that this wff is of the form $\exists \mathsf{z}\mathsf{R}(\mathsf{x}, \mathsf{y}, \mathsf{z})$ with just one initial unbounded quantifier, where $\mathsf{R}(\mathsf{x}, \mathsf{y}, \mathsf{z})$ is Δ_0. (We'll generalize in a moment.)

R will, of course, express some three-place relation R, where $\mathsf{R}(\overline{\mathsf{m}}, \overline{\mathsf{n}}, \overline{\mathsf{o}})$ is true just when $Rmno$. And then $f(m) = n$ just when $\exists z Rmnz$.

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 $g(x)$ puts a ceiling c on the numbers we need to search through before finding a pair n, o such that $Rxno$ is true. And $h(x, c)$ looks for a number n under the ceiling c such that for some o also under that ceiling $Rxno$ is true. Which means, of course, that

$$f(x) = h(x, g(x))$$

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,

$$\begin{aligned}\mathsf{G}(\mathsf{x}, \mathsf{y}) =_{\text{def}} {} & (\exists \mathsf{u} \leq \mathsf{y})(\exists \mathsf{v} \leq \mathsf{y})\mathsf{R}(\mathsf{x}, \mathsf{u}, \mathsf{v}) \\ & \wedge (\forall \mathsf{w} \leq \mathsf{y})[\mathsf{w} \neq \mathsf{y} \rightarrow \neg(\exists \mathsf{u} \leq \mathsf{w})(\exists \mathsf{v} \leq \mathsf{w})\mathsf{R}(\mathsf{x}, \mathsf{u}, \mathsf{v})] \\ \mathsf{H}(\mathsf{x}, \mathsf{y}, \mathsf{z}) =_{\text{def}} {} & [(\exists \mathsf{v} \leq \mathsf{y})\mathsf{R}(\mathsf{x}, \mathsf{z}, \mathsf{v}) \wedge \neg(\exists \mathsf{u} \leq \mathsf{z})(\exists \mathsf{v} \leq \mathsf{y})(\mathsf{u} \neq \mathsf{z} \wedge \mathsf{R}(\mathsf{x}, \mathsf{u}, \mathsf{v}))] \\ & \vee [\neg(\exists \mathsf{v} \leq \mathsf{y})\mathsf{R}(\mathsf{x}, \mathsf{z}, \mathsf{v}) \wedge \mathsf{z} = 0]\end{aligned}$$

and those wffs are evidently Δ_0.

We now just need to generalize to the case where we are dealing with a many-place function $f(\vec{x}) = y$, and where the strictly Σ_1 wff which expresses f has the form $\exists \vec{\mathsf{z}}\mathsf{R}(\vec{\mathsf{x}}, \mathsf{y}, \vec{\mathsf{z}})$, perhaps with more than one existential quantifier before the Δ_0 core $\mathsf{R}(\mathsf{x}, \mathsf{y}, \vec{\mathsf{z}})$. So $\vec{\mathsf{z}}$ is perhaps a whole array of variables $\mathsf{z}_1, \mathsf{z}_2, \ldots \mathsf{z}_\mathsf{n}$, and $\exists \vec{\mathsf{z}}$ unpacks as a whole bunch of corresponding existential quantifiers, $\exists \mathsf{z}_1 \exists \mathsf{z}_2 \ldots \exists \mathsf{z}_\mathsf{n}$.

Well, the generalization is easy: we just proceed in the same way, putting $\vec{x}$ for x, $\vec{v}$ for v, $\vec{\mathsf{x}}$ for x, and $\vec{\mathsf{v}}$ for v, and then reading e.g. $(\exists \vec{\mathsf{v}} \leq \mathsf{y})$ in the obvious way, as short for a bunch of bounded quantifiers $(\exists \mathsf{v}_1 \leq \mathsf{y})(\exists \mathsf{v}_2 \leq \mathsf{y}) \ldots \exists(\mathsf{v}_\mathsf{n} \leq \mathsf{y})$.

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 was 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.

[2] Credit where credit is due: I learnt this neat dodge from (Boolos et al., 2002, p. 206).

We'll take the simplest case (again, generalizing the argument is easy, and is left as an exercise). So suppose that f is a one-place Σ_1 function, and suppose

$$f(x) = h(x, g(x))$$

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 $\widetilde{\mathsf{G}}(\mathsf{x}, \mathsf{y})$ and $\widetilde{\mathsf{H}}(\mathsf{x}, \mathsf{y}, \mathsf{z})$ 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 $\mathsf{F}(\mathsf{x}, \mathsf{y}) =_{\text{def}} \exists \mathsf{z}(\widetilde{\mathsf{G}}(\mathsf{x}, \mathsf{z}) \wedge \widetilde{\mathsf{H}}(\mathsf{x}, \mathsf{z}, \mathsf{y}))$.

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:

$$\forall \mathsf{y}(\widetilde{\mathsf{G}}(\overline{\mathsf{m}}, \mathsf{y}) \leftrightarrow \mathsf{y} = \overline{\mathsf{o}})$$
$$\forall \mathsf{z}(\widetilde{\mathsf{H}}(\overline{\mathsf{m}}, \overline{\mathsf{o}}, \mathsf{z}) \leftrightarrow \mathsf{z} = \overline{\mathsf{n}})$$

But by elementary logic, those two imply

$$\forall \mathsf{y}(\exists \mathsf{z}(\widetilde{\mathsf{G}}(\overline{\mathsf{m}}, \mathsf{z}) \wedge \widetilde{\mathsf{H}}(\overline{\mathsf{m}}, \mathsf{z}, \mathsf{y})) \leftrightarrow \mathsf{y} = \overline{\mathsf{n}})$$

But that means we have

$$\text{If } f(m) = n, \text{ then } \forall \mathsf{y}(\mathsf{F}(\overline{\mathsf{m}}, \mathsf{y}) \leftrightarrow \mathsf{y} = \overline{\mathsf{n}})$$

Which shows that F captures f.

Finishing the proof We have shown that every Σ_1 function is a composition of two Δ_0 functions, and that such a composition of functions can be captured in Q (by a Σ_1 wff). Therefore Q can capture every Σ_1 function (by a Σ_1 wff). Which – given the result of Section 10.2 – entails that Q can capture *as a function* every Σ_1 function (and indeed do *that* by a Σ_1 wff). Which gives us Theorem 13*. ⊠

In sum, we've given a recipe for taking a wff φ which expresses f and turning it into a wff φ' which captures f as a function. (Though note that φ' will also still express f. For since Q is sound, wffs that capture express it too, by the same argument as at the end of Section 4.6.)

11.3 L_A can express all p.r. functions: starting the proof

That's one major theorem under our belt. 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 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 Theorem 14 it is enough 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 $\mathsf{Z(x, y)} =_{\text{def}} (\mathsf{x} = \mathsf{x} \wedge \mathsf{y} = \mathsf{0})$.

Finally, the three-place identity function $I^3_2(x, y, z) = y$, to take just one example, is expressed by the wff $\mathsf{I}^3_2\mathsf{(x, y, z, u)} =_{\text{def}} (\mathsf{x} = \mathsf{x} \wedge \mathsf{y} = \mathsf{u} \wedge \mathsf{z} = \mathsf{z})$. 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 $\mathsf{G(x, y)}$ and $\mathsf{H(x, y)}$ respectively. Then, the function $f(x) = h(g(x))$ is expressed by the wff $\exists\mathsf{z(G(x, z)} \wedge \mathsf{H(z, y))}$. Other cases where g and/or h are multi-place functions can be handled similarly. ⊠

Starting the proof 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 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?

11.4 The idea of a β-function

Let's pause the proof 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 9.8; for this function returns the exponent of the prime number π_i in the unique 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]

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.

[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).

[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. As so often, we have a trade-off. We are making life harder for ourselves at this point by working with Q/PArather than $\mathsf{Q}^+/\mathsf{PA}^+$. The pay-off is that our eventual incompleteness theorems show that there is no complete theory even for the basic arithmetic of L_A truths.

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]

But now reflect that the concept of a remainder on division can be elementarily defined in terms of multiplication and addition. Thus consider the following open wff:

[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 $rm(c, d_0)$, $rm(c, d_1)$, $rm(c, d_2)$, ..., $rm(c, d_n)$. 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.

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). ⊠

$$\mathsf{B}(\mathsf{c},\mathsf{d},\mathsf{i},\mathsf{y}) =_{\text{def}} (\exists \mathsf{u} \leq \mathsf{c})[\mathsf{c} = \{\mathsf{S}(\mathsf{d} \times \mathsf{Si}) \times \mathsf{u}\} + \mathsf{y} \ \wedge \ \mathsf{y} \leq (\mathsf{d} \times \mathsf{Si})]$$

This, as we want, expresses our Gödelian β-function in L_A (and shows that it is a Δ_0 function).

11.5 L_A can express all p.r. functions: finishing the proof

Continuing the proof 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:

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.

Concluding the proof for (3) We need to show that we can we can use the same β-function trick and prove (3): *any* function f defined by recursion from functions g and h which are already expressible in L_A is also expressible in L_A.

So 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 establish (1), (2) and (3) from the start of Section 11.3. 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 too!

11.6 The p.r. functions are Σ_1

(a) 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 *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 a wff which expresses the p.r. function f by the dodge of *recapitulating the function's p.r. definition*. As we track through this definition we write down a wff to express the function defined at each stage. Initial functions are expressed by Δ_0 (hence Σ_1) wffs. Compositions are expressed using existential quantifiers applied to previous wffs expressing the functions we want to compose: so compositions don't take us beyond what can be expressed with Σ_1 wffs. A function f defined by recursion from g and h is expressed by a wff like (C*), again built up from the previous wffs expressing g and h. 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.

Since that little trick of dragging quantifiers around doesn't disguise things very much, we can still say that the series of Σ_1 wffs recapitulating some function's full p.r. definition yields, at the end of the process, a *perspicuous* way of expressing that function (i.e. a way of expressing it from which we can recover a definition of the function expressed).

(b) Now we'll check the key claims (but by all means skip the rest of this section with the fiddly details: we are just joining up the dots). Following the same proof strategy that we laid out in Section 11.3, 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 in Section 7.6, 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 $\mathsf{G}(\mathsf{x},\mathsf{y}) = \exists\mathsf{u}\mathsf{G}'(\mathsf{x},\mathsf{y},\mathsf{u})$ and $\mathsf{H}(\mathsf{x},\mathsf{y}) = \exists\mathsf{v}\mathsf{H}'(\mathsf{x},\mathsf{y},\mathsf{v})$ 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 $\exists\mathsf{z}(\mathsf{G}(\mathsf{x},\mathsf{z}) \wedge \mathsf{H}(\mathsf{z},\mathsf{y}))$, i.e. $\exists\mathsf{z}(\exists\mathsf{u}\mathsf{G}'(\mathsf{x},\mathsf{z},\mathsf{u}) \wedge \exists\mathsf{v}\mathsf{H}'(\mathsf{z},\mathsf{y},\mathsf{v}))$. But that's equivalent to $\exists\mathsf{z}\exists\mathsf{u}\exists\mathsf{v}(\mathsf{G}'(\mathsf{x},\mathsf{z},\mathsf{u}) \wedge \mathsf{H}'(\mathsf{z},\mathsf{y},\mathsf{v}))$ 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′) First, let's introduce the simple 'quantifier-shift' trick we need. Take the sample wff

i. $(\forall\mathsf{u} \leq \overline{\mathsf{n}})\exists\mathsf{v}\mathsf{K}(\mathsf{u},\mathsf{v},\overline{\mathsf{m}})$.

If (i) is true of some number m, then for each $u \leq n$, there is a corresponding witness w_u which makes $\mathsf{K}(\mathsf{u},\overline{\mathsf{w_u}},\overline{\mathsf{m}})$ true. Now, take w to be the largest of those $n+1$ witnesses w_u. Then $(\forall\mathsf{u} \leq \overline{\mathsf{n}})(\exists\mathsf{x} \leq \overline{\mathsf{w}})\mathsf{K}(\mathsf{u},\mathsf{x},\overline{\mathsf{m}})$ is true. And therefore

ii. $\exists\mathsf{v}(\forall\mathsf{u} \leq \overline{\mathsf{n}})(\exists\mathsf{x} \leq \mathsf{v})\mathsf{K}(\mathsf{u},\mathsf{x},\overline{\mathsf{m}})$

is true. Hence if (i) is true, so is (ii); but obviously if (ii) is true, so is (i). Hence we can find a wff which expresses the same as (i) – because it is true of a number m just when (i) is – but which brings the *unbounded* existential quantifier $\exists\mathsf{v}$

out in front of the bounded universal quantifier ($\forall \mathsf{u} \leq \overline{\mathsf{n}}$), leaving behind – as it were, as its shadow – a new *bounded* existential quantifier.[6]

Obviously, the quantifier shift trick generalizes. Suppose, then, that 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

$$\begin{array}{ll} \text{C}^* & \exists \mathsf{c} \exists \mathsf{d}\{\exists \mathsf{k}[\mathsf{B}(\mathsf{c}, \mathsf{d}, 0, \mathsf{k}) \wedge \mathsf{G}(\vec{\mathsf{x}}, \mathsf{w})] \wedge \\ & \quad (\forall \mathsf{u} \leq \mathsf{y})[\mathsf{u} \neq \mathsf{y} \rightarrow \exists \mathsf{v} \exists \mathsf{w}\{(\mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{u}, \mathsf{v}) \wedge \mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{Su}, \mathsf{w})) \wedge \mathsf{H}(\vec{\mathsf{x}}, \mathsf{u}, \mathsf{v}, \mathsf{w})\}] \wedge \\ & \quad \mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{y}, \mathsf{z})\} \end{array}$$

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*) because it is satisfied by exactly the same numbers. Hence a function defined by recursion from Σ_1 functions is itself Σ_1. ⊠

In sum, putting together (1) to (3) we have proved, as we wanted, that a function defined by recursion from Σ_1 functions is itself Σ_1.

11.7 The adequacy theorem

We can now simply put together the two Big Results which we've established in Section 11.2 and the last section:

Theorem 13* Q *can capture any* Σ_1 *function as a function.*

Theorem 15 *Every p.r. function is* Σ_1,

These immediately entail the 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.*

[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 one 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. We won't keep on repeating this sort of caveat.

[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, have the absolutely key result that the β-function trick can be used to express any p.r. function in L_A.

This implies that Q can also capture every p.r. property and relation. That's because a property is p.r. if it has a p.r. characteristic function; and this function, being p.r., can be captured in Q. But by the trivial result we noted in Section 10.2, 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 means that PA is p.r. adequate too. Indeed, because it contains enough induction, PA will be strongly p.r. adequate (i.e. can fully define every p.r. function as a function).

11.8 Canonically capturing

Note that we have proved our Beautiful Big Theorem *constructively*. The claim is that, for any p.r. function f, there exists a wff φ by which Q can capture it. But our proof doesn't just make a bald existence claim: it tells us how to construct a particular φ that does the job, at least once we are presented with a p.r. definition for f.

Here's the recipe again, in two stages. First stage. Take a p.r. definition for f. This definition proceeds by defining a whole series of functions ending with f, where each function in the series is either an initial function, or is defined from earlier functions by composition or recursion. Now, for each of those functions, we can write down a corresponding wff which expresses it. This will be a Δ_0 (and hence Σ_1) wff in the case of any initial function. When a function is defined by composition from two functions earlier in the series, we write down a strictly Σ_1 wff built by existential quantification from the two wffs which express those earlier functions (as in the proof of $(2')$ in Section 11.6). When a function is defined by recursion from two previous functions in the series, we write down a strictly Σ_1 wff built on the model of (C**) from the two wffs which express the previous functions. So in this way, we eventually get to a strictly Σ_1 wff φ which expresses f, essentially by recapitulating a p.r. definition f.

Now the second stage. We use the trick explained in Section 11.2 to convert φ into a corresponding wff φ' which not only expresses but captures f. This wff still reflects all the details of f's p.r. definition (and we can recover that definition from the wff, revealing the function in question to indeed be p.r.). Let's say that such a wff, built in the way described, *canonically captures* f.

Three remarks. (i) There isn't a unique way of canonically capturing a given p.r. function f in Q. Boringly we can relabel variables, and shift the order of conjuncts; more importantly, there will always be alternative ways of giving p.r. definitions for f (if only by including redundant detours). (ii) There will be innumerable non-canonical ways for capturing any f: just recall the remark in Section 4.5, (b), pointing out in effect that if $\varphi(\vec{\mathsf{x}}, \mathsf{y})$ captures f in some theory, so does $\varphi(\vec{\mathsf{x}}, \mathsf{y}) \wedge \theta$, for *any* theorem θ. (iii) By the point we made at the end of Section 11.2, a wff that does canonically capture f in Q (and hence in PA) will also express it.

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 used one elegant 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. We'll say rather more about the historical context in a later Interlude on Hilbert's Programme. But 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 8.6, 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 and claim that Socrates has some instances.

[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.

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 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.

Having proposed a paradox-blocking theory of types as their 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.

This isn't the place to review the details and differences of the Frege-Russell constructions, their successes and failures. Still, for those who haven't seriously

[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.

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, '$\exists!$' is the familiar uniqueness quantifier (see Section **??**, 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 $x = 0 \vee x = 1$. 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 $x = 0 \vee x = 1 \vee x = 2$. 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 on a *philosophical* objection at this point. I just hope to have said enough to give you a hint of how Frege and the authors of *Principia* could think that there was a possible enterprise here. But now enter Gödel, with a devastating *formal* objection.

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 that there are wffs φ in effect belonging to L_A, the language 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 philosophical 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 even of basic arithmetic. And similarly, there are basic arithmetical propositions which are 'formally undecidable' in ZF set theory.

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]

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 $Prf(m, n)$ 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 **??**)

2. Since $Prf(m, n)$ 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, is true if and only if it is 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 **??**)

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 theory that contains the weaker theory Q is incompletable. (Chapter **??**)

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 constitutes a computable function (a general theory which didn't become settled until the later 1930s).

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.

Ackermann, W., 1928. On Hilbert's construction of the real numbers. In van Heijenoort 1967, pp. 495–507.

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.

Bezboruah, A. and Shepherdson, J. C., 1976. Gödel's second incompleteness theorem for Q. *Journal of Symbolic Logic*, 41: 503–512.

Black, R., 2000. Proving Church's Thesis. *Philosophia Mathematica*, 8: 244–258.

Boolos, G., 1993. *The Logic of Provability*. Cambridge: Cambridge University Press.

Boolos, G., Burgess, J., and Jeffrey, R., 2002. *Computability and Logic*. Cambridge: Cambridge University Press, 4th edn.

Bridges, D. S., 1994. *Computability: A Mathematical Sketchbook*. New York: Springer-Verlag.

Büchi, J. R., 1962. Turing machines and the *Entscheidungsproblem*. *Mathematische Annalen*, 148: 201–213.

Buss, S. R., 1998. Proof theory of arithmetic. In S. R. Buss (ed.), *Handbook of Proof Theory*, pp. 79–147. Elsevier Science B.V.

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.

Cohen, D. E., 1987. *Computability and Logic*. Chichester: Ellis Horwood.

Cooper, S. B., 2004. *Computability Theory*. Boca Raton, Florida: Chapman and Hall/CRC.

Copeland, B. J. (ed.), 2004. *The Essential Turing*. Oxford: Clarendon Press.

Craig, W., 1953. On axiomatizability within a system. *Journal of Symbolic Logic*, 18: 30–32.

Curry, H. B., 1942. The inconsistency of certain formal logics. *Journal of Symbolic Logic*, 2: 115–117.

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, 2nd edn.

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.

Fitch, F. B., 1952. *Symbolic Logic*. New York: Roland Press.

Folina, J., 1998. Church's Thesis: prelude to a proof. *Philosophia Mathematica*, 6: 302–323.

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.

Gentzen, G., 1935. Untersuchungen über das logische schliessen. *Mathematische Zeitschrift*, 39: 176–210, 405–431. Translated as 'Investigations into logical deduction', in *The Collected Papers of Gerhard Gentzen*, edited by M. E. Szabo, Amsterdam: North Holland Publishing Co., 1969.

Gödel, K., 1929. On the completeness of the calculus of logic. In Gödel 1986, pp. 60–101.

Gödel, K., 1931. On formally undecidable propositions of *Principia Mathematica* and related systems I. In Gödel 1986, pp. 144–195.

Gödel, K., 1932. Consistency and completeness. In Gödel 1986, pp. 234–237.

Gödel, K., 1934. On undecidable propositions of formal mathematical systems. In Gödel 1986, pp. 346–371.

Gödel, K., 1936. On the length of proofs. In Gödel 1986, pp. 396–399.

Gödel, K., 1972. Some remarks on the undecidability results. In Gödel 1990, pp. 305–306.

Gödel, K., 1986. *Collected Works, Vol. 1: Publications 1929–1936*. New York and Oxford: Oxford University Press.

Gödel, K., 1990. *Collected Works, Vol. 2: Publications 1938–1974*. 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.

Henkin, L., 1952. A problem concerning provability. *Journal of Symbolic Logic*, 17: 160.

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.

Hilbert, D. and Bernays, P., 1939. *Grundlagen der Mathematik, Vol II*. 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.

Keefe, R. and Smith, P. (eds.), 1999. *Vagueness: A Reader*. Cambridge, MA: MIT Press.

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., 1936c. A note on recursive functions. *Bulletin of the American Mathematical Society*, 42: 544–546.

Kleene, S. C., 1938. On notation for ordinal numbers. *Journal of Symbolic Logic*, 3: 150–155.

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.

Kolmogorov, A. N. and Uspenskii, V. A., 1963. On the definition of an algorithm. *American Mathematical Society Translations*, 29: 217–245.

Kreisel, G., 1965. Mathematical logic. In T. L. Saaty (ed.), *Lectures on Modern Mathematics*, vol. III, pp. 95–195. New York: John Wiley.

Kreisel, G., 1972. Informal rigour and completeness proofs. In I. Lakatos (ed.), *Problems in the Philosophy of Mathematics*. Amsterdam: North-Holland.

Kreisel, G., 1980. Kurt gödel. *Biographical Memoirs of Fellows of the Royal Society*, 26: 149–224.

Kretzmann, N. and Stump, E., 1988. *The Cambridge Translations of Medieval Philosophical Texts: Vol. 1, Logic and the Philosophy of Language*. Cambridge University Press.

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.

Löb, M. H., 1955. Solution of a problem of Leon Henkin. *Journal of Symbolic Logic*, 20: 115–118.

Mendelson, E., 1990. Second thoughts about Church's thesis and mathematical proofs. *Journal of Philosophy*, 87: 225–233.

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.

Mostowski, A., 1966. *Thirty Years of Foundational Studies*. Oxford: Basil Blackwell.

Nelson, R. J., 1987. Church's Thesis and cognitive science. *Notre Dame Journal of Formal Logic*, 28: 581–614.

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.

Péter, R., 1935. Konstruktion nichtrekursiver Funktionen. *Mathematische Annalen*, pp. 42–60.

Péter, R., 1951. *Rekursive Funktionen*. Budapest: Akadémiai Kiadó.

Péter, R., 1967. *Recursive Functions*. New York: Academic Press.

Potter, M., 2004. *Set Theory and its Philosophy*. Oxford: Oxford University Press.

Quine, W. V., 1940. *Mathematical Logic*. Cambridge, MA: Harvard University Press.

Rice, H. G., 1953. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74: 358–366.

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.

Schönhage, A., 1970. Universelle Turing Speicherung. In J. Dörr and G. Hotz (eds.), *Automatentheorie und Formal Sprachen*. Mannheim: Bibliogr. Institut.

Schönhage, A., 1980. Storage modification machines. *SIAM Journal on Computing*, 9: 490–508.

Shepherdson, J. C. and Sturgis, H. C., 1963. Computability of recursive functions. *Journal of the Association for Computing Machinery*, 10: 217–255.

Shoenfield, J. R., 1967. *Mathematical Logic*. Reading, MA: Addison-Wesley.

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.

Smoryński, C., 1977. The incompleteness theorems. In J. Barwise (ed.), *Handbook of Mathematical Logic*, pp. 821–865. Amsterdam: North-Holland.

Smoryński, C., 1985. *Self-Reference and Modal Logic*. New York: Springer-Verlag.

Smullyan, R. M., 1992. *Gödel's Incompleteness Theorems*. Oxford: Oxford University Press.

Steen, S. W. P., 1972. *Mathematical Logic*. Cambridge: Cambridge University Press.

Tarski, A., 1933. *Pojecie Prawdy w Jezykach Nauk Dedukcyjnych*. Warsaw. Translated into English in Tarksi 1956, pp. 152–278.

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 Dalen, D., 1994. *Logic and Structure*. Berlin: Springer-Verlag, 3rd edn.

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.

Wang, H., 1974. *From Mathematics to Philosophy.* Routledge and Kegan Paul.

Wittgenstein, L., 1989. *Wittgenstein's 1939 Cambridge Lectures on the Foundations of Mathematics.* C. Diamond, ed. Chicago: Chicago University Press.