CAMBRIDGE INTRODUCTIONS TO PHILOSOPHY

An Introduction to Gödel's Theorems

PETER SMITH

CAMBRIDGE www.cambridge.org/9780521857840

An Introduction to Gödel's Theorems

In 1931, the young Kurt Gödel published his First Incompleteness Theorem, which tells us that, for any sufficiently rich theory of arithmetic, there are some arithmetical truths the theory cannot prove. This remarkable result is among the most intriguing (and most misunderstood) in logic. Gödel also outlined an equally significant Second Incompleteness Theorem. How are these Theorems established, and why do they matter? Peter Smith answers these questions by presenting an unusual variety of proofs for the First Theorem, showing how to prove the Second Theorem, and exploring a family of related results (including some not easily available elsewhere). The formal explanations are interwoven with discussions of the wider significance of the two Theorems. This book will be accessible to philosophy students with a limited formal background. It is equally suitable for mathematics students taking a first course in mathematical logic.

Peter Smith is Lecturer in Philosophy at the University of Cambridge. His books include *Explaining Chaos* (1998) and *An Introduction to Formal Logic* (2003), and he is a former editor of the journal of *Analysis*.

An Introduction to Gödel's Theorems

Peter Smith
University of Cambridge

CAMBRIDGE
UNIVERSITY PRESS

CAMBRIDGE UNIVERSITY PRESS
Cambridge, New York, Melbourne, Madrid, Cape Town, Singapore, São Paulo

Cambridge University Press
The Edinburgh Building, Cambridge CB2 8RU, UK
Published in the United States of America by Cambridge University Press, New York
www.cambridge.org
Information on this title: www.cambridge.org/9780521857840

First published in print format 2007

ISBN-13 978-0-511-35096-2 eBook (MyiLibrary)
ISBN-10 0-511-35096-1 eBook (MyiLibrary)

ISBN-13 978-0-521-85784-0 hardback
ISBN-10 0-521-85784-8 hardback

For Patsy, as ever

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 – necessarily briefly – 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 general audience (the best of those is Franzén, 2005), but who find the rather forbidding presentations in classic texts in mathematical logic (like Mendelson, 1997) 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; don't try to browse through too quickly. We do cover a lot of ground in quite a bit of detail, and new ideas often come thick and fast. Take things slowly!

Theorems are numbered in the standard way, so Theorem 16.2 is the second theorem in Chapter 16. The distinction between a 'proof', a 'proof sketch' and (occasionally) a 'sketch of a proof sketch' is blurry. The end of a proof or proof sketch is marked by ☒.

I assume only a modest amount of background in logic. So we can't cover, for example, material that presupposes a serious knowledge of model theory; which means that we don't discuss (say) model-theoretic arguments for incompleteness. That's a pity. But there's a sequel planned for enthusiasts who want to know about such matters.

I originally intended to write a rather shorter book, leaving many 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 textbooks 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 are exercises on the book's website at www.godelbook.net.

As we go through, there is also a small 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 'μ-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.

My colleague Michael Potter has been an inspiring presence since I returned to Cambridge. Many thanks are due to him and to all those who have very kindly given me comments on parts of various drafts, including the late Torkel Franzén, Tim Button, Luca Incurvati, Jeffrey Ketland, Aatu Koskensilta, Christopher Leary, Mary Leng, Toby Ord, Alex Paseau, Jacob Plotkin, José F. Ruiz, Kevin Scharp, Hartley Slater, and Tim Storer. I should especially mention Richard Zach, whose comments at two different stages were particularly extensive and particularly helpful.

But my greatest debt is to Patsy Wilson-Smith, without whose continuing love and support this book would never have been written.

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, $m < n$ iff there is a number $k \neq 0$ such that $m + k = n$. m is a factor of n iff $0 < m$ and there is some number k such that $0 < k$ and $m \times k = n$. m is even iff it has 2 as a factor. m is prime iff $1 < m$ and m's only factors are 1 and itself. And so on.[1]

Using our basic and/or defined concepts, we can then make various general claims about the arithmetic of addition and multiplication. There are familiar 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

[1] 'Iff' is, of course, the standard logicians' shorthand for 'if and only if'.

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 school-room 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 φ is true; or its negation $\neg\varphi$ is deducible from the axioms, and so φ is false.[2] We may not, of course, actually stumble on a proof one way or the other: but the idea is that such a proof always exists, 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 that jargon, the suggestion we are considering is this: we should be able to specify a reasonably simple bundle of true axioms which, together with some logic, give us a *complete* theory of basic arithmetic: we could in principle use the theory to prove the truth or falsity of any claim about addition and/or multiplication (or at least, any claim we can state using quantifiers like 'for all', connectives like 'if' and 'not', and identity). And if that's right, truth in basic arithmetic could just be equated with provability in this complete theory.

It is tempting to say more. For what will the axioms of basic arithmetic look like? Here's one candidate: 'For every natural number, there's a unique next one'. This is evidently true: but evident *how*? Is it that we have some special and rather mysterious faculty of mathematical intuition which allows us just to *'see'* that this axiom is true? Or can we avoid an appeal to intuition? Maybe the axiom is evidently true because it is some kind of definitional triviality. Perhaps 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

[2] 'Normal proof' is vague, and soon 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.

unique successor. And, plausibly, other candidate axioms are similarly true by definition (or are logically derivable from definitions).

If those tempting thoughts are right – if the truths of basic arithmetic all flow deductively from logic plus definitionally true axioms – then true arithmetical claims would be simply *analytic* in the philosophers' sense.[3] And this so-called 'logicist' view would then give us a very neat explanation of the special certainty and the necessary truth of correct claims of basic arithmetic.

1.2 Incompleteness

But now, in headline terms, *Gödel's First Incompleteness Theorem shows that the entirely natural idea that we can completely 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, e.g. that T is consistent) that neither G_T nor $\neg\mathsf{G}_T$ can be derived in T, and yet (ii) we can also recognize that, at least if T *is* consistent, G_T will be true.

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 from which we can deduce everything else.

How does Gödel show this in his great 1931 paper which presents the Incompleteness Theorems? 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 ends with '1' if she is an undergraduate student and with '2' if she is a graduate; and so on and so forth. More excitingly, we can use numbers and numerical propositions to encode facts about theories, e.g. facts about what can be derived in a theory T.[4] And

[3] Thus Gottlob Frege, writing in his wonderful *Grundlagen der Arithmetik*, 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.' (Frege, 1884, p. 4)

[4] 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 does not mean that φ is proved in the sense that it is 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 with 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 is the logician's notion which is in play.

what Gödel did is find a general method that enabled him to take any theory T strong enough to capture a modest amount of basic arithmetic and construct a corresponding arithmetical sentence G_T which encodes the claim 'The sentence G_T itself is unprovable in theory T'. So G_T is true if and only if T can't prove it.

Suppose that T has true axioms and a reliably truth-preserving deductive logic. Then everything T proves must be true, i.e. T is a *sound* theory. 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, because T only proves truths. 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: the official First Theorem shows, for a start, that T's mere consistency is enough to guarantee that a suitably constructed G_T is true-but-unprovable-in-T.

To repeat: the sentence G_T encodes the claim that that very sentence is unprovable. But doesn't this make G_T uncomfortably reminiscent of the Liar sentence 'This very sentence is false' (which is false if it is true, and true if it is false)? You might well wonder whether Gödel's argument doesn't lead to a cousin of the Liar paradox rather than to a theorem. But not so. As we will soon see, there is nothing at all suspect or paradoxical about Gödel's First Theorem as a technical result about formal axiomatized systems (a result which we can in any case prove without appeal to 'self-referential' sentences).

'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 that ten people are taking the logic course. But if there's no relevant 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.

Finally, 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 does not just affect theories of basic arithmetic. Consider 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 all the sets so far constructed. We get the sequence

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

This sequence has the structure of the natural numbers. We can pick out 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. Moreover, any standard set theory can define this sequence. So 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; adding a simple routine for translating the results for this fragment into the familiar language of basic arithmetic would then give us a complete theory of arithmetic. Hence, by Gödel's First Incompleteness Theorem, there cannot be a negation-complete set theory.

The point evidently generalizes: any axiomatized mathematical theory T that can define (an analogue of) the natural-number sequence and replicate enough of the basic arithmetic of addition and multiplication must be incomplete and incompletable.[5]

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 all flows deductively from a simple bunch of 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?

[5] We return to this point more carefully in Section 18.2.

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 theory T, then we'll be able to derive $0 \neq 1$.[6] So if T *also* proves $0 = 1$, it is inconsistent. Conversely, if T is inconsistent, then – since we can derive anything in an inconsistent theory[7] – it can prove $0 = 1$. But we said that we can use numerical propositions to encode facts about what can be derived in T. So there will in particular be a *numerical* consistency sentence Con_T that encodes the claim that we can't derive $0 = 1$ in T, i.e. encodes in a natural way the claim that T is consistent.

We know, however, that there is a numerical proposition which encodes the claim that G_T is unprovable: we have already said that it is G_T itself.

So this means that (part of) the conclusion of Gödel's First Theorem, namely the claim that if T is consistent, then G_T is unprovable, can *itself* be encoded by a numerical proposition, namely $\mathsf{Con}_T \to \mathsf{G}_T$. And 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 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 $\mathsf{Con}_T \to \mathsf{G}_T$ 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.[8]

[6] We'll allow ourselves to abbreviate expressions of the form $\neg\sigma = \tau$ as $\sigma \neq \tau$.

[7] 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.

[8] That *is* rough. The Second Theorem shows that T can't prove Con_T, which is certainly *one* natural way of expressing T's consistency inside T. But couldn't there perhaps be some

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 entails 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 *itself* to be 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 '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 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 which use only '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 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 introduce some concepts relating specifically to axiomatized theories of arithmetic. Then in Chapters 5 and 6 we prove a pair of neat and relatively easy results – namely that any sound and 'sufficiently expressive' axiomatized theory of arithmetic, and likewise any consistent and 'sufficiently strong' axiomatized theory, is negation incomplete. For reasons that we'll explain, these informal results fall some way short of Gödel's own First Incompleteness Theorem. But they do provide a very nice introduction to some key ideas that we'll be developing more formally in the ensuing chapters.

other sentence of T, Con'_T, which also in some good sense expresses T's consistency, where T doesn't prove $\mathsf{Con}'_T \to \mathsf{G}_T$ but *does* prove Con'_T? We'll return to this question in Sections 24.5 and 27.2.

2 Decidability and enumerability

This chapter briskly introduces a number of concepts – mostly related to the idea of computability – that we'll need in the next few chapters. Later in the book, we'll return to some of these ideas and give sharper, technical, treatments of them. But for present purposes, informal intuitive presentations are enough.

2.1 Functions

We'd better start, however, by very quickly reviewing some standard jargon and notation for talking about functions, since functions will feature so prominently in what follows. For simplicity, we'll focus here on one-place functions (it will be obvious how to generalize definitions to cover many-place functions).

Our concern will be with *total* functions $f\colon \Delta \to \Gamma$, i.e. with functions which map *every* element x of the *domain* Δ to exactly one corresponding value $f(x)$ in the set Γ.[1] We then say

i. The *range* of a function $f\colon \Delta \to \Gamma$ is $\{f(x) \mid x \in \Delta\}$, i.e. the set of elements in Γ that are values of f for arguments in Δ.

ii. 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.)

iii. A function $f\colon \Delta \to \Gamma$ is *injective* iff f maps different elements of Δ to different elements of Γ – i.e. if $x \neq y$ then $f(x) \neq f(y)$. (If you prefer that in English, you can say that such a function is *one-to-one*.)

iv. A function $f\colon \Delta \to \Gamma$ is *bijective* if it is both surjective and injective. (In English again, f is then a *one-one correspondence* between Δ and Γ.)

2.2 Effective decidability, effective computability

(a) Familiar school-room arithmetic routines (e.g. for testing whether a number is prime) give us ways of *effectively deciding* whether some property holds. Other

[1] For wider mathematical purposes, the more general idea of a *partial* function becomes essential. This is a mapping f which is not necessarily defined for all elements of its domain (for an obvious example, consider the reciprocal function $1/x$ for rational numbers, which is not defined for $x = 0$). However, we won't need to say much about partial functions in this book, and hence – by default – plain 'function' will henceforth always mean 'total function'.

routines (e.g. for squaring a number or finding the highest common factor of two numbers) give us ways of *effectively computing* the value of a function.

What is meant by talking of *effective* procedures? Well, we are trying to sharpen the otherwise rather vague, intuitive, notion of a computation. And the core idea is that an effective procedure involves executing an *algorithm* which *successfully terminates*.

Here, an algorithm is a set 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). More carefully, executing an algorithm (i) involves an entirely determinate sequence of discrete step-by-small-step procedures (where each small step is readily executable by a very limited calculating agent or machine). (ii) There isn't any room left for the exercise of imagination or intuition or fallible human judgement. Further, in order to execute the procedures, (iii) we don't have to resort to outside 'oracles' (i.e. independent sources of information), and (iv) we don't have to resort to random methods (coin tosses). Such algorithmic procedures can be followed by a dumb computer. Indeed, it is natural to turn this observation into a first shot at an informal definition:

> An algorithmic procedure is one that a suitably programmed computer can execute.

But plainly, if an algorithmic procedure is actually to decide whether some property holds or actually to compute a function, more is required. It needs to terminate after a finite number of steps and deliver a result!

So, putting these ideas together, we can give two interrelated rough definitions:

> A property/relation is *effectively decidable* iff there is an algorithmic procedure that a suitably programmed computer could use to decide, in a finite number of steps, whether the property/relation applies in any given case.
>
> A total function is *effectively computable* iff there is an algorithmic procedure that a suitably programmed computer could use for calculating, in a finite number of steps, the value of the function for any given argument.[2]

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

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

[2]For more about how to relate these two definitions via the notion of a 'characteristic function', see Section 11.6.

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 executing its algorithmic procedure for centuries.

Still, we are cheerfully going to abstract from all these 'merely practical' considerations of size and speed – which is why we said nothing about them in explaining what we mean by effective procedures. In other words, we will count a function as being 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 limitations of time and memory. 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 longer to compute whether a given number has it than we have time 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 a fair challenge. And modern computer science has much to say about grades of computational complexity and 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, e.g. about what can't be algorithmically decided. By working with a very weak 'in principle' notion of what is required for being decidable, our impossibility results will be correspondingly very strong – they won't depend on any 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 that some problems can't be mechanically decided, even on the most generous understanding of that idea.

(c) 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 that it doesn't (at least, once we are dealing with devices of a certain degree of complexity, which can act as 'general purpose' computers). 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 31). 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.[3] The details don't matter here and now. What does matter is that exactly the same functions are computable by algorithms written for Mark I Turing machines, by algorithms written for Mark II machines, and by algorithms written for register machines, despite their different architectures.

Indeed, *all* the detailed 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.

(d) Let's put that last claim a bit more slowly, taking it in two stages. First stage: there's a Big Result about the mutual equivalence of various different proposed ways of refining the idea of algorithmically computing *numerical* functions and deciding *numerical* properties. That's a formal mathematical result. But it supports the conjecture which Turing famously makes in his classic paper published in 1936:

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

As we'll see, however, Turing machines are rather horrible to work with (essentially, you have to program them at the level of 'machine code'). So you might want to think instead in terms of the numerical functions which are computable – at least when we abstract from limitations of time and memory space – on a modern general purpose computer, using programs written in your favourite general purpose language, C++ perhaps. Then Turing's Thesis is provably equivalent to this: the numerical functions that are effectively computable in the informal

[3]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).

sense are just those functions that are in principle computable using algorithms written in C++.

Turing's Thesis – we'll further explore its content in Chapter 34 – correlates an informal notion with a sharp technical analysis. So you might think it isn't the sort of thing for which we can strictly speaking give a *proof* (though see Chapter 35). But be that as it may. Certainly, after some seventy years, no successful challenge to Turing's Thesis has ever been mounted. Which means that we can continue to talk informally about effectively computable numerical functions and effectively decidable properties of numbers, and be very confident that we are referring to fully determinate classes.

(e) Second stage: 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, they are determinate enough for our purposes. For 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 a computation dealing with sufficiently determinate and distinguishable Xs can always 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 fullest generality. For the purposes of this book, the non-numerical computations we are most 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, 15.1, 15.2). So the question e.g. whether a certain property of formulae is a decidable one can be translated quite uncontentiously into the question whether a corresponding numerical property is a decidable one. Given Turing's Thesis that it is quite determinate what counts as a decidable property of *numbers*, it then follows that it is quite determinate what counts as a decidable property of *formal expressions* (and similarly for properties of sequences of expressions).

2.3 Enumerable sets

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

Suppose, then, that Σ is some set of items: its members might be numbers, strings of symbols, proofs, computer programs or whatever. Then, as a first rough shot, we'll say:

> Σ is *enumerable* if its members can – at least in principle – be listed off in some order (a 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!

Of course, if we are to talk of 'listing off' elements of Σ, then we really need to be thinking of these elements either as being things that themselves can be written down (like strings of symbols), or as having standard representations that can be written down (in the way that natural numbers have numerals which denote them). But that condition will be satisfied in the cases that most interest us in this book.

For the pernickety, however, we can give a more rigorous definition that doesn't presuppose that we have a way of writing down the members of Σ. In fact, we can do this in a number of equivalent ways: we'll give just one. As usual, we'll use '$\mathbb{N}$' to denote the set of all natural numbers (and '$\mathbb{N}^2$' for the set of pairs of numbers, etc.). Then here's our more official definition:

> The set Σ is enumerable iff either Σ is empty or else there is a surjective function $f\colon \mathbb{N} \to \Sigma$ (so Σ is the range of f: we can say that such a function enumerates Σ).

It is easy to see that this comes to the same as our original informal definition (in the cases we are interested in, when we *can* write down the members of Σ).

Proof Both definitions trivially cover the case where Σ is empty. So concentrate on the non-empty cases.

Suppose we have a list – possibly infinite, possibly repetitious – of all the members of Σ in some order. Count off the members of the list from zero, and define the function f as follows: $f(n) =$ the n-th member of the list, if the list goes on that far, or $f(n) = f(0)$ otherwise. Then $f\colon \mathbb{N} \to \Sigma$ is a surjection.

Suppose conversely that $f\colon \mathbb{N} \to \Sigma$ is a surjection. Then, if we successively evaluate f for the arguments 0, 1, 2, ... in turn, we get a corresponding list of values $f(0), f(1), f(2), \ldots$ which by hypothesis we can write down and which contains all the elements of Σ, with repetitions allowed. ⊠

Every set of natural numbers, finite or infinite, is enumerable (its members can be put into a list). However, *Cantor's Theorem*[4] tells us that

Theorem 2.1 *There are infinite sets that are not enumerable.*

Proof Consider the set $\mathbb{B}$ of infinite binary strings s, i.e. the set of unending strings like '0110001010011 . . .'. There's obviously an infinite number of different ones. Suppose, for reductio, that there is an enumerating function which maps the natural numbers onto the binary strings as follows:

$$\begin{array}{ccl} 0 & \rightarrow & s_0 : \underline{0}110001010011\ldots \\ 1 & \rightarrow & s_1 : 1\underline{1}00101001101\ldots \\ 2 & \rightarrow & s_2 : 11\underline{0}0101100001\ldots \\ 3 & \rightarrow & s_3 : 000\underline{1}111010101\ldots \\ 4 & \rightarrow & s_4 : 1101\underline{1}11011101\ldots \\ \ldots & \ldots & \end{array}$$

Go down the diagonal, taking the n-th digit of the n-th string s_n (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 s_0, the initial string in our original enumeration, in the first place; it differs from the next string s_1 in the next place; and so on. So our diagonal construction defines a new string s that differs from each of the s_j, contradicting the assumption that our enumerating function is 'onto', i.e. that it enumerates *all* the binary strings. So $\mathbb{B}$ is infinite, but not enumerable. In a word, it is *indenumerable*. ⊠

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

First, an infinite binary string $s = b_0b_1b_2\ldots$ 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 in the interval $[0, 1]$ can't be enumerated (and hence we can't enumerate *all* the reals either).

Second, an infinite binary string $s = b_0b_1b_2\ldots$ can also be thought of as characterizing a corresponding set of natural numbers Σ, where $n \in \Sigma$ if $b_n = 0$ and $n \notin \Sigma$ if $b_n = 1$. So, for example, the string s_0 above corresponds to the set of numbers $\{0, 3, 4, 5, 7, 9, 10, \ldots\}$. Our theorem is therefore equivalent to the result that the set of *sets* of natural numbers can't be enumerated. Given any enumeration of sets of numbers Σ_0, Σ_1, Σ_2 . . . , then 'going down the diagonal' defines the set K such that $k \in K$ iff $k \in \Sigma_k$. And 'flipping' gives us $\overline{K}$, the complement of K, which differs from each Σ_k, so can't be on the enumeration.

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

[4]Georg Cantor first established this key result in Cantor (1874), using the Bolzano-Weierstrass theorem. The neater 'diagonal argument' first appears in Cantor (1891).

to the result that the set of *functions* $f\colon \mathbb{N} \to \{0, 1\}$ can't be enumerated. Put in terms of functions, the trick in the proof is to suppose that these functions *can* be enumerated $f_0, f_1, f_2, \ldots$, define another function by 'going down the diagonal and flipping digits', i.e. define $\delta(n) = f_n(n) + 1 \mod 2$, and then note that this diagonal function δ can't be on the list after all.

2.4 Effective enumerability

We said: a non-empty set Σ is enumerable so long as there is some way of listing its elements, or – equivalently – so long as there is a function $f\colon \mathbb{N} \to \Sigma$ which enumerates it. Note that this definition does not require that the listing can be done 'mechanically'. In other words, the enumerating function f here can be any arbitrary correlation of numbers with elements of Σ (so long as it is 'onto'); f need not even be finitely specifiable, let alone be a 'nice' effectively computable function.

So let's now add a further definition:

> A set Σ is *effectively enumerable* if an (idealized) computer could be programmed to generate a list of its members such that any member will eventually be mentioned – the list may be empty, or have no end, and may contain repetitions, so long as any item in the set eventually makes an appearance.

Again, that informal characterization will do for most purposes. But more officially, we will say:

> The set Σ is effectively enumerable iff either Σ is empty or else there is an *effectively computable* function that enumerates it.[5]

Reflect that a function $f\colon \mathbb{N} \to \Sigma$ can only be evaluated by a computer if the elements of Σ are the sort of things that a computer can handle representations for (i.e., if elements of Σ can be listed). So these two definitions can also readily be seen to be equivalent, by a minor tweak of the argument as before.

Proof Again both definitions trivially cover the case where Σ is empty. So concentrate on the non-empty cases.

Suppose the algorithm Π lists the members $\sigma \in \Sigma$, and o is the first of them listed. Then the following describes a slightly more complex algorithm Π' which takes numerical inputs. Given input n, run Π for n steps: if at that step Π outputs some σ, then Π' also outputs σ; otherwise it outputs o. This algorithm evidently computes a numerical function f whose range is the whole of Σ.

[5] 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 entirely stable: some writers use 'enumerable' to mean *effectively* enumerable, and use e.g. 'denumerable' for the wider notion of enumerability.

Suppose conversely that f enumerates the members of Σ. We can simply run the algorithm that computes f on the inputs 0, 1, 2, ... in turn to get a way of mechanically listing all the members of Σ. ⊠

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. For a simple example of an effectively enumerable *infinite* set, imagine an algorithm that takes the natural numbers one at a time in order and applies the well-known mechanical test for being prime, and lists the successes: this procedure generates a never-ending list on which every prime will eventually appear – so the primes are effectively enumerable.

We'll soon see, however, that there are important cases of infinite sets which are enumerable but which *can't* be effectively enumerated (see Theorem 5.5).

2.5 Effectively enumerating pairs of numbers

Here's another quick example of an effectively enumerable set, for later use.

> **Theorem 2.2** *The set of ordered pairs of natural numbers $\langle i, j\rangle$ is effectively enumerable.*

Proof The idea is simple. We just arrange the ordered pairs in a systematic array and start zig-zagging through them, for example like this:

$$\begin{array}{cccccc}
\langle 0,0\rangle & \rightarrow & \langle 0,1\rangle & & \langle 0,2\rangle & \rightarrow \langle 0,3\rangle \ \ldots \\
& \swarrow & & \nearrow & & \swarrow \\
\langle 1,0\rangle & & \langle 1,1\rangle & & \langle 1,2\rangle & \langle 1,3\rangle \ \ldots \\
\downarrow & \nearrow & & \swarrow & & \\
\langle 2,0\rangle & & \langle 2,1\rangle & & \langle 2,2\rangle & \langle 2,3\rangle \ \ldots \\
& \swarrow & & & & \\
\langle 3,0\rangle & & \langle 3,1\rangle & & \langle 3,2\rangle & \langle 3,3\rangle \ \ldots \\
\downarrow & & & & & \\
\end{array}$$

This procedure is entirely mechanical, runs through all the pairs, and evidently defines a bijection $f\colon \mathbb{N} \rightarrow \mathbb{N}^2$, a one-one correspondence mapping n to the n-th pair on the zig-zag path. ⊠

If you have a taste for trivial arithmetical puzzles, you can explicitly define a computable function $pair(i, j)$ which tells you the position n of the pair $\langle i, j\rangle$ in the zig-zag effective enumeration. You can likewise write down two computable functions $fst(n)$ and $snd(n)$ which return, respectively the first member i and the second member j of the n-th pair in the zig-zag enumeration. But we can leave that as an exercise.

3 Axiomatized formal theories

Gödel's Incompleteness Theorems tell us about the limits of 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 exploring the idea and proves some elementary results about axiomatized formal theories in general.

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 fully formalized versions. 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 modern formal system of logic – albeit with a horrible notation – 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.

These days, quite a few mathematical theories are presented axiomatically in a more formal way from the very outset. For example, set theories are typically presented by laying down some basic axioms expressed in a logical language 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.

Now, even the most tough-minded mathematics texts which explore axiomatized theories are written in an informal mix of ordinary language and mathematical symbolism. Proofs are rarely spelt out in every formal detail, and so 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'.[2] But still, it *is* absolutely essential for good mathematics to achieve precision and to avoid the use of unexamined inference rules or unacknowledged assumptions. So, putting together the logician's

[2] '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)

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.

Note, we are not saying that mathematicians ought really always to work inside fully formalized 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.[3] 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 observations don't affect our present 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 very brisk, as we don't want to get bogged down in details, and most of the ideas here should be very familiar.[4] Our main concern is to emphasize points about decidability.

(a) 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 ideally supposed to be just that, i.e. *proofs* with content, which show things to be true. Agreed, we'll often be very interested in certain 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 $\mathcal{L}$ is a syntactically defined system of expressions and $\mathcal{I}$ gives the intended interpretation of these expressions.

[3] 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.

[4] If the ideas *aren't* familiar, any logic textbook will help – see e.g. Leary (2000, Ch. 1).

(b) Start with the syntactic component $\mathcal{L}$. We can assume that this is based on a finite alphabet of symbols.[5] And we 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 to specify which symbols or finite strings of symbols make up $\mathcal{L}$'s *non-logical vocabulary*, e.g. the individual constants (names), predicates, and function-signs. Finally, we need 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; but of course, our main interest will be in the $\mathcal{L}$-*sentences*, i.e. closed wffs without variables dangling free.

Let's make 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 do not want it to be a disputable matter whether a given symbol or cluster of symbols is e.g. a constant or one-place predicate of a system $\mathcal{L}$. Nor, crucially, do we want disputes about whether a given string of symbols is an $\mathcal{L}$-wff. And we don't want disputes either about what is a closed $\mathcal{L}$-wff, i.e. an $\mathcal{L}$-sentence.

So, whatever the details, for a properly formalized syntax $\mathcal{L}$, 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 an $\mathcal{L}$-wff/$\mathcal{L}$-sentence. It goes almost without saying that the formal languages familiar from elementary logic have this feature.

(c) Let's move on, then, to the interpretation $\mathcal{I}$. Our prime aim is to fix the content of each closed $\mathcal{L}$-wff, i.e. each $\mathcal{L}$-sentence. And standardly, we fix the content of formal sentences by giving truth-conditions, i.e. by saying what it would take for a given sentence to be true. However, we can't, in the general case, do this just by giving a list associating $\mathcal{L}$-sentences with truth-conditions (for the simple reason that there will be an unlimited number of sentences). We'll therefore 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 the simplest sort of case; we can again assume that this is all broadly familiar from elementary logic. Suppose, then, that $\mathcal{L}$ has the usual syntax of a first-order language (for the moment, without identity or function signs).[6] A standard interpretation $\mathcal{I}$ will start by fixing the *domain* of quantification, will assign *values* in the domain to constants, and will give *satisfaction conditions* for predicates. For example, perhaps

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

[6] 'First-order' means that the quantifiers run over the objects in the domain: compare 'second-order' logic which also allows a second sort of quantifier that runs over properties of those objects. See Section 22.1.

> The domain of quantification is the set of people;
> the value of 'm' is Socrates;
> the value of 'n' is Plato;
> something satisfies 'F' iff it is wise;
> 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. 'Fm' is true just in case the value of 'm' satisfies 'F' (i.e. iff Socrates is wise); 'Lmn' is true just in case the ordered pair ⟨value of 'm', value of 'n'⟩ satisfies 'L' (i.e. iff Socrates loves Plato); and so on.[7]

Now we need to deal with the logical vocabulary. First, there are the usual rules for assigning truth-conditions to sentences built up out of simpler ones using the propositional connectives.

That leaves the quantifiers to deal with. Take the existential case. Here's one way of telling the story. Intuitively, if the quantifier is to range over people, then '$\exists\mathsf{x}\mathsf{Fx}$' 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 have specified 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 *extend* the interpretation $\mathcal{I}$ to $\mathcal{I}^+$ by assigning the new name κ a value in the domain in such a way that $\varphi(\kappa)$ is true on the expanded interpretation $\mathcal{I}^+$. Similarly for universal quantifiers.

Given the aims of formalization, a compositional semantics needs to yield an unambiguous truth-condition for each sentence (and do it in an effective way). The usual accounts of the semantics of the standard formal languages of logic have this feature of effectively generating unique readings for sentences.[8]

[7] Note that something satisfies 'F' according to $\mathcal{I}$ iff it is wise, hence iff it is in the set of wise people. Call that set associated with 'F' its *extension*. Then 'Fm' is true on interpretation $\mathcal{I}$ iff the value of 'm' 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.

[8] 'But what about wffs with free variables? Or what comes to much the same, wffs with 'parameters', as used in natural deduction proofs? You haven't said how an interpretation $\mathcal{I}$ assigns *them* truth-conditions!' True. *But we don't need to.* We only need to think of such wffs as getting truth-values on interpretations $\mathcal{I}^+$ which *extend* our original interpretation $\mathcal{I}$ by assigning references to the variables/parameters, in effect treating the wffs in question as sentences involving temporary names. For more, see e.g. Tennant (1978, pp. 29, 71–74).

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 *(non-logical) axioms*, i.e. as the fundamental non-logical assumptions of our theory. Of course, we'll normally want these axioms to be sentences which are true on interpretation: but that needn't be 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 to be found among 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.[9]

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 which use logical axioms 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 overall system we adopt. We will predominantly be working with some version of standard first-order logic with identity. But whatever system we adopt, we need to be able to specify it in a way which enables us to settle, without room for dispute, what counts as a well-formed derivation.

In other words, *we require the property of being a well-formed proof from*

[9] 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 $\forall x\varphi(x)$ where 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: that's an impossible task. 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, even if we don't put any contingent bound on the length of proofs.

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.[10]

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 30.5).

(c) In the case of an axiomatized formal theory T, it is decidable which wffs are T's axioms (whether non-logical axioms, or logical axioms if it has any), and it is decidable which arrays of wffs conform to the derivation rules of T's proof system. It will therefore be decidable which arrays of wffs are axiomatic T-proofs – i.e. which arrays are properly constructed proofs, all of whose premisses are indeed T-axioms. So, to summarize:

> 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/sentence 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.

3.4 More definitions

Here are six more standard definitions, specifically to do with theories:

i. Given a derivation of the *sentence* 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: $T \vdash \varphi$.

[10] 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).

ii. A theory T is *sound* iff every theorem of T is true (i.e. true on the interpretation built into T's language). Soundness is, of course, normally a matter of having true axioms and a truth-preserving proof system.

iii. A theory T is *decidable* iff the property of being a theorem of T is an effectively decidable property – i.e. iff there is a mechanical procedure for determining, for any given sentence φ of T's language, whether $T \vdash \varphi$.

iv. Assume now that T has a standard negation connective '$\neg$'. A theory T *decides* the sentence 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, $T \vdash \neg\varphi$.

v. 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 $T \vdash \neg\varphi$).

vi. T is *inconsistent* iff for some sentence φ, we have both $T \vdash \varphi$ and $T \vdash \neg\varphi$.

Note, it is convenient to restrict the theorems to the derivable *sentences*, wffs without free variables; but nothing really hangs on this.

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 interpreted propositional atoms 'p', 'q', 'r' together with all the wffs that can be constructed from 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 {'$\neg\mathsf{p}$', 'q', '$\neg\mathsf{r}$'}. 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 a proof from the given axioms. Both theories are consistent. Moreover, both are decidable theories; 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.

By contrast, 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.

Our 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*.[11]

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 3.1** *If T is an axiomatized formal theory then (i) the set of wffs of T, (i′) the set of sentences of T, (ii) the set of proofs 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, put the symbols of the finite alphabet into some order. Then start by listing all the strings of length 1, followed by all those of length 2 in 'alphabetical order', followed by all those of length 3 in 'alphabetical order', and so on and so forth.

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 mechanical procedures together, as we ploddingly enumerate all the possible strings we can

[11] 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 Σ and any sentence φ,

if $\Sigma \vDash \varphi$ then $\Sigma \vdash \varphi$,

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 the topic of his (First) Incompleteness Theorem: the semantic completeness of a proof system for quantificational logic is one thing, the negation incompleteness of certain theories of arithmetic quite a different thing.

throw away the non-wffs that turn up, leaving us with an effective enumeration of all the wffs. ⊠

Proof sketch for (i′) As for (i), replacing 'wff' by 'sentence'. ⊠

Proof sketch for (ii) Assume that T-proofs are linear sequences of wffs. Just as we can effectively enumerate all the possible wffs, so we can effectively enumerate all the possible finite sequences of wffs in some 'alphabetical order'. One brute-force way is to start effectively 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 these proof sequences from the other sequences of wffs, to give us an effective enumeration of all the possible proofs. (If T-proofs are more complex arrays of wffs – as in tree systems – then the construction of an effective enumeration of the arrays needs to be correspondingly more complex: but the core proof-idea remains the same.) ⊠

Proof sketch for (iii) Start effectively enumerating the well-constructed proofs again. But this time, just record their conclusions when they pass the mechanical test for being closed sentences. This mechanically generated list now contains all and only the theorems of the theory. ⊠

Just one comment on 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:[12]

> **Theorem 3.2** *Any consistent, axiomatized, negation-complete formal theory T is decidable.*

Proof We know from Theorem 3.1 that there's an algorithm for effectively enumerating the theorems of T. So start effectively listing the theorems. Let φ

[12] By the way, it is trivial that an *inconsistent* axiomatized theory with a classical logic is decidable. For if T is inconsistent, every wff of T's language is a theorem 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.

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: φ is a theorem. If on the other hand $\neg\varphi$ is produced, stop the enumeration, for 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 2.2). We might have to twiddle our thumbs for an immense time before one of φ or $\neg\varphi$ turns 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.

4 Capturing numerical properties

The previous chapter 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 is consistent and satisfies some other 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 might 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 inevitably makes expressions of informal everyday arithmetic and their formal counterparts look very similar. And while context alone should 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 our ordinary type-face (mostly in *italics*) for informal mathematics, and using a sans-serif font for expressions in our formal languages. Thus compare ...

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

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 entails 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. But note that Greek letters will never belong to our formal languages themselves: these symbols belong to logicians' augmented English.

So what exactly is going on when we are talking about a formal language 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 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 which contain both formal and meta-linguistic 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 normal) 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 in order to fend off possible misunderstandings.)

Now, just as one and the same thing can be picked out by two co-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 '... is exactly divisible 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, so let's not delay over them.

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. $(\mathsf{a} + \mathsf{b})$ and $(\mathsf{a} \times \mathsf{b})$ rather than $+(\mathsf{a}, \mathsf{b})$ and $\times(\mathsf{a}, \mathsf{b})$.

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 $\mathsf{0}$, $\mathsf{0}'$, $\mathsf{0}''$, $\mathsf{0}'''$, ... (We won't be using that notation in this book: still, I'll mostly avoid using the prime symbol for other purposes when there could be any possibility of a casually 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. In particular, numerals count as closed 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 the *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. And given the standard reading of the identity relation, it is immediate that

> 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: thus

> 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 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 $\mathbb{N}$, and a sentence of the form $\exists\xi\varphi(\xi)$ is true on $\mathcal{I}_A$ just if there is some number in the domain which we can dub with a 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 the intended domain of quantification $\mathbb{N}$ 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.

4.4 A quick remark about truth

The semantics $\mathcal{I}_A$ entails that the sentence $(1 + 2) = 3$, i.e. $(\mathsf{S0} + \mathsf{SS0}) = \mathsf{SSS0}$, is true just so long as one plus two is three. Likewise the sentence $\exists\mathsf{v}\,\mathsf{4} = (\mathsf{v} \times 2)$, i.e. $\exists\mathsf{v}\,\mathsf{SSSS0} = (\mathsf{v} \times \mathsf{SS0})$, 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 take an arithmetical theory T built in the language L_A, and construct a sentence G_T which turns out to be 'true but unprovable-in-T'. And while the sentence in question is a bit exotic, there is nothing in the least exotic about the notion of truth being applied to it 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.

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.[2] Such common-or-garden arithmetic claims, they aver, presuppose the existence of numbers as mysterious kinds of objects in some Platonic heaven, and they doubt the literal existence of such things. In the view of many of these philosophers, arithmetical entities should be thought of as useful *fictions*: and, at least when we are on our very best behaviour, we really ought to claim 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'.

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

4.5 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. $\exists \mathsf{v}(2 \times \mathsf{v} = \overline{\mathsf{n}})$.

For $n = 4$, for example, this unpacks into '$\exists \mathsf{v}(\mathsf{SS0} \times \mathsf{v} = \mathsf{SSSS0})$'. Abbreviate such a wff by $\psi(\overline{\mathsf{n}})$. Then 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. So consider the corresponding open wff[3] with one free variable

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

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 iff 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 wffs of the type

2. $(\overline{\mathsf{n}} \neq 1 \ \wedge \ \forall \mathsf{u} \forall \mathsf{v}(\mathsf{u} \times \mathsf{v} = \overline{\mathsf{n}} \ \rightarrow \ (\mathsf{u} = 1 \ \vee \ \mathsf{v} = 1)))$

(where we use $\alpha \neq \beta$ for $\neg\alpha = \beta$). Abbreviate such a wff by $\chi(\overline{\mathsf{n}})$. Then $\chi(\overline{\mathsf{n}})$ 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.

The corresponding open wff

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

is therefore satisfied by exactly the prime numbers. In other words, $\chi(\mathsf{x})$ 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:

[3] Usage varies: in this book, an open wff is one which isn't closed, i.e. which has at least one free variable, though it might have other bound variables.

> 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 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]

(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.

[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 technical and philosophical reasons for preferring the gappy notation to express properties and relations. However, it is the default informal mathematical practice to prefer to use complete expressions with free variables rather than expressions with place-holders which mark gaps; sticking to this practice therefore makes for a more familiar-looking notation and hence aids readability. (Trust me! – I did at one stage try writing this book systematically using Greek letters as place-holders, and some passages looked quite unnecessarily repellent.)

Still, there's a wrinkle. Just once, in Section 12.4, 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.

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. However, the intuitive notion is murky and notoriously difficult to analyse (even if we can usually recognize wffs which 'express the right meaning' when we meet them). By contrast, our notion is sharply defined, and it will serve us perfectly well for most purposes.

4.6 Capturing numerical properties and relations

(a) Of course, we don't merely want various properties and relations 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 or stand in which relations (more carefully: we want formal derivations which will be proofs in the intuitive sense if can we take it that the axioms are indeed secure truths).

Now, it is a banal observation that to establish facts about *individual* numbers typically requires less sophisticated proof-techniques than proving general truths about *all* numbers. So let's focus 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 general 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 $T \vdash \neg\varphi(\overline{\mathsf{n}})$.

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 the more intuitive sense of proving wffs with the right meaning. But again the intuitive notion is murky, and our sharply defined notion will serve us very well for many purposes (though we will later also be introducing a notion of 'canonically capturing', see Section 13.8).

4.7 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 12). But there are some irritating minor complications in that case, so let's not delay over it here: instead, we'll immediately press on in the next couple of chapters 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 wff $\varphi(\mathsf{x})$ satisfying our conditions for capture. But I'm unapologetic: '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 *axioms* and *proof system*.[6]

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 in T's language but not capturable by T (see e.g. Section 21.4). However, there is a link in the other direction. Suppose T is a *sound* theory of arithmetic, i.e. one whose theorems are all true on the given arithmetic interpretation of its language. 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 immediately entails that *if* $\varphi(\mathsf{x})$ *captures* P *in the sound theory* T*, then* $\varphi(\mathsf{x})$ *expresses* P.

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

Gödel originally talked of a numerical relation being 'decidable' (*entscheidungsdefinit*) when it is captured by an arithmetical wff (Gödel, 1931, p. 176). As later alternatives to our 'captures' we find 'numeralwise expresses' (Kleene 1952, p. 195; Fisher 1982, p. 112), and also simply 'expresses'(!) again (Mendelson, 1997, p. 170), 'formally defines' (Tourlakis, 2003, p. 180) and plain 'defines' (Boolos et al., 2002, p. 207). At least 'binumerate' – (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 The truths of arithmetic

In Chapter 3, we proved that the theorems of any properly axiomatized theory – and hence, in particular, the theorems of any properly axiomatized arithmetic – *can* be effectively enumerated. In this chapter, we prove by contrast that the truths of any sufficiently expressive arithmetic language *can't* be effectively enumerated (we will explain in just a moment what 'sufficiently expressive' means).

Suppose then that T is a properly axiomatized theory with a sufficiently expressive language. Since T is axiomatized, its theorems can be effectively enumerated. Since T's language is sufficiently expressive, the truths of its language can't be effectively enumerated. Hence the theorems and the truths can't be the same: either some T-theorems aren't truths, or some truths aren't T-theorems. Let's concentrate on sound theories whose theorems *are* all true. Then for any sound axiomatized theory T which is sufficiently expressive, there will be truths which aren't T-theorems. Let φ be such an unprovable truth: then $\neg\varphi$ will be false, so that too will be unprovable in our sound theory T. Hence T must be negation incomplete.

So much for the headline news. The rest of this chapter fills in the details.

5.1 Sufficiently expressive languages

Recall: a two-place relation R is effectively decidable iff there is an algorithmic procedure that decides whether Rmn, for any given m and n (Section 2.2). And a relation R can be expressed in language L iff there is an open L-wff φ such that $\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is true iff Rmn (Section 4.5).

We will now say that

> An interpreted formal language L is *sufficiently expressive* iff (i) it can express every effectively decidable two-place numerical relation, and (ii) it can express quantifications over numbers.

As we've just announced, we are going to show that sound axiomatized theories with sufficiently expressive languages can't be negation complete.

Of course, that wouldn't be an interesting result if a theory's having a sufficiently expressive language were a peculiarly tough condition to meet. But it isn't. Much later in this book, in Section 30.1, we'll show that even L_A, the language of basic arithmetic, is sufficiently expressive (in fact, L_A can express every decidable numerical property and every decidable numerical relation with *any* number of places). However, we can't yet establish this claim about L_A: doing that would obviously require having a general theory of decidable properties

and relations, and so far we haven't got one. For the moment, then, we'll just *assume* that theories with sufficiently expressive languages are worth thinking about, and see what follows.[1]

5.2 More about effectively enumerable sets

We are going to show that the set of truths of a sufficiently expressive language is not effectively enumerable. To do this, we need three initial theorems about effectively enumerable sets.

(a) We start with an easy warm-up exercise. Recall: a set of natural numbers W is effectively enumerable iff it is either empty or there is an effectively computable function f which enumerates it.[2] Then,

> **Theorem 5.1** *If W is an effectively enumerable set of numbers, then there is some effectively decidable numerical relation R such that $n \in W$ if and only if $\exists x Rxn$.*

Proof The theorem hold trivially when W is empty (simply choose a suitable R that is never satisfied). Suppose, then, that the effectively computable function f enumerates W. That means the values $f(0), f(1), f(2), \ldots$ give us all and only the members of W. So $n \in W$ iff $\exists x f(x) = n$. We now just define Rmn to hold iff $f(m) = n$. Evidently, we can decide whether Rmn obtains just be evaluating $f(m)$ and checking whether the result is indeed n – and both steps are, by hypothesis, algorithmically computable. So we are done. ⊠

(b) Our second, more substantial, result gives us another way of characterizing effectively enumerable sets of numbers. Recall we said that an algorithmic procedure is one that a suitably programmed computer can execute (abstracting from limitations of time or memory space). To fix ideas, concentrate for the rest of this section on algorithms written in your favourite general purpose language, C++ as it might be. And here's a definition: the *numerical domain* of such an algorithm Π is the set of natural numbers n such that when the algorithm Π is

[1] Though perhaps we can do a *bit* better than mere assumption even at this point, for consider the following line of argument. If R *is* decidable, that means there is a computer program Π which can be used to tell us whether Rmn. In other words, Rmn if and only if the decision program Π with inputs m and n terminates with a 'yes' verdict. But now imagine that we use numerical coding – procedures for associating programs and descriptions of the computer's memory states etc. with numbers – to encode claims about when a computer program gives a 'yes' verdict. There will then be a statement $\mathsf{R}(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ in an arithmetical language L which encodes the claim that Π gives a 'yes' verdict on inputs m and n, so $\mathsf{R}(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is true just when Rmn. Hence R expresses R, and L is sufficiently expressive. Or at least, this will be so if L is rich enough to have the resources to code up descriptions of the behaviour of programs. However, it should seem fairly plausible that such coding needn't take *very* much arithmetical language – as we'll indeed confirm in later chapters.

[2] By the way, the $W/W_e/K$ notation we use in this section for various sets of numbers is fairly standard.

applied to the number n as input, then Π will eventually terminate and deliver some output (in principle, given world enough and time).

Many algorithms aren't even intended to apply to (single) numbers, so will have an empty numerical domain.[3] While for some successful numerical algorithms, the numerical domain will be $\mathbb{N}$, the whole set of natural numbers. But whatever the algorithm, its domain will always be an effectively enumerable set. For we have the following result:

> **Theorem 5.2** *W is an effectively enumerable set of numbers if and only if it is the numerical domain of some algorithm Π.*

Proof of the 'only if' direction Suppose W *is* an effectively enumerable set of numbers. Then either (i) W is empty, or (ii) there is an effectively computable total function f which enumerates W. In case (i), choose a Π that never produces any output and we are done. In case (ii) there is some algorithm which computes the enumerating function f, so we can construct the following composite algorithmic procedure Π. Given number n as input, compute the values of $f(0), f(1), f(2), \ldots$ in turn: keep going on and on unless and until one of those values turns out to be n – and as soon as does, stop and output the number 0. Then the numerical domain of Π is obviously W (for Π will terminate just when fed an $n \in W$). ⊠

Proof of the 'if' direction Suppose that W is the numerical domain of some algorithm Π. If W is empty, then trivially it is effectively enumerable.

So suppose W isn't empty and o is some member of it. And now recall the pairing functions we introduced in Section 2.5. Each possible pair of numbers $\langle i, j \rangle$ gets effectively correlated one-to-one with a number n, and there are computable functions $fst(n)$ and $snd(n)$ which return, respectively, the first member i and the second member j of the n-th pair. Using these, define the new algorithm Π' as follows. Given input n, compute $i = fst(n)$, and $j = snd(n)$. Then run Π on input i for j steps (we defined algorithms as involving discrete step-by-step procedures, so we can number the steps). If Π on input i has halted with some output by step j, then Π' outputs i. Otherwise Π' outputs the default value o.

As n increases, this procedure Π' evaluates Π for every input i, for any number of steps j; it outputs just those arguments i for which Π eventually delivers some output. So Π' computes a total function whose range is the whole of Π's numerical domain W. Hence W is indeed effectively enumerable. ⊠

(c) We supposed we were working within some general-purpose programming language like C++ in which we can regiment instructions for performing (the equivalent of) any algorithm that is supposed to operate on numbers.[4] So by

[3] And if you have ever done any programming, you'll have learnt the hard way that the numerical domain of an algorithm which *is* supposed to apply to numbers is only too often the empty set again, because your program crashes and doesn't produce any output!

[4] Of course, there is an assumption here that there *are* general purpose languages, apt for regimenting instructions for all kinds of numerical algorithms. Much later, when we talk more

effectively listing off in some 'alphabetical order' all the possible strings of symbols that obey the syntax for being a series of well-formed program instructions in that language, we can effectively list off versions of all the possible algorithms $\Pi_0, \Pi_1, \Pi_2, \ldots$ (most of them will be useless algorithms which 'crash', of course). Let the numerical domain of Π_e be W_e. Then, by our last theorem, every effectively enumerable set is W_e for some index e.

It is now easy to prove

> **Theorem 5.3** *There is an effectively enumerable set of numbers K such that its complement $\overline{K}$ is not effectively enumerable.*

For we just need to put $K =_{\text{def}} \{e \mid e \in W_e\}$, as we'll now show.[5]

Proof that $\overline{K}$ is not effectively enumerable For any e, by definition $e \in \overline{K}$ if and only if $e \notin W_e$. Hence, $\overline{K}$ cannot be identical to any of the W_e (since e is in one but not the other), so $\overline{K}$ isn't one of the effectively enumerable sets (since the W_e are all of them). ⊠

Proof that K is effectively enumerable Since $\overline{K}$ is not effectively enumerable, $\overline{K}$ isn't the whole of $\mathbb{N}$ (for that is trivially effectively enumerable!), so K isn't empty. So let o be some member of K. Now consider the effective procedure Π'' defined as follows. Given input n, compute $i = fst(n)$, and $j = snd(n)$. Then find Π_i, and run it on input i for j steps. If Π_i on input i has halted with some output by step j, then Π'' outputs i. Otherwise Π'' outputs the default value o. As n increases, this procedure runs through all pairs of values i, j: so the output of Π'' is the set of *all* numbers i such that i is in the numerical domain of Π_i, i.e. is the set of i such that $i \in W_i$, i.e. is K. So K is effectively enumerable. ⊠

5.3 The truths of arithmetic are not effectively enumerable

Given the easy results in the last section, it immediate follows that

> **Theorem 5.4** *The set of truths of a sufficiently expressive arithmetical language L is not effectively enumerable.*

Proof Take the argument in stages. (i) Theorem 5.3 tells us that there a set K which is effectively enumerable but whose complement isn't. Theorem 5.1 then entails that there is a decidable relation R such that $n \in K$ iff $\exists x R(x, n)$.

(ii) Since R is decidable, then in any given sufficiently expressive arithmetical language L there will be some formal expression of L which expresses R: let's abbreviate that particular formal expression $\mathsf{R}(\mathsf{x}, \mathsf{y})$.

about Turing's Thesis, we'll give strong warrants for this claim. But familiarity with modern computing practice should make our assumption seem entirely reasonable.

[5] Compare the analogous construction in our comments on Cantor's Theorem 2.1.

(iii) Then Rmn just when $\mathsf{R}(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is true; and so $\exists x R(x, n)$ just when $\exists \mathsf{x} \mathsf{R}(\mathsf{x}, \overline{\mathsf{n}})$ is true (we assumed, recall, that a sufficiently expressive language can express quantifications over numbers).

(iv) So from (i) and (iii) we have

$n \in K$ if and only if $\exists \mathsf{x} \mathsf{R}(\mathsf{x}, \overline{\mathsf{n}})$ is true; i.e.,
$n \in \overline{K}$ if and only if $\neg \exists \mathsf{x} \mathsf{R}(\mathsf{x}, \overline{\mathsf{n}})$ is true.

(v) Now concentrate on that second way of putting the equivalence. And suppose for a moment that the set $\mathcal{T}$ of truths of arithmetic expressible in L *is* effectively enumerable. Then, given a description of the expression R, we could run through the supposed effective enumeration of $\mathcal{T}$, and whenever we come across a truth of the type $\neg \exists \mathsf{x} \mathsf{R}(\mathsf{x}, \overline{\mathsf{n}})$ – and it will be effectively decidable if a wff has a particular syntactic form – list the number n. That procedure would give us an effectively generated list of all the members of $\overline{K}$.

(vi) But by hypothesis $\overline{K}$ is *not* effectively enumerable. So $\mathcal{T}$ can't be effectively enumerable after all. Which is what we wanted to show. ⊠

Which is, I hope you agree, a beautiful result. Yet it was simply proved. We have just taken a couple of pretty elementary observations about effectively enumerable sets (Theorems 5.1 and 5.2). Then we defined a pair of sets $K/\overline{K}$. That's the same kind of little trick that was involved in Cantor's proof. It was then very easy to show Theorem 5.3. And these elementary ingredients immediately give us Theorem 5.4.

A comment. The informal idea of (all) 'the truths of arithmetic' is no doubt not a sharp one: what exactly should we include? But however we refine it, presumably we want the truths of arithmetic, broadly construed, to include at least the truths about the nice, decidable, properties and relations of numbers. So in our jargon, the truths of arithmetic, on any plausible sharpening of that idea, should be the truths of a language which is sufficiently expressive. Hence our theorem warrants the informal claim expressed in the title of this section: the truths of arithmetic can't be effectively enumerated.

5.4 Incompleteness

We now derive three easy consequences of Theorem 5.4, working up to our first version of an incompleteness theorem.

(a) We first prove a result which we announced at the very end of Chapter 2:

Theorem 5.5 *There are enumerable sets which are not effectively enumerable.*

Proof Theorem 5.3 tells us that there are sets of natural numbers which are not effectively enumerable: but all sets of natural numbers are enumerable.

Less abstractly, we've just shown in Theorem 5.4 that the set of truths (i.e. true sentences) of a sufficiently expressive formal language L is not effectively

enumerable. But again that set *is* enumerable. For we can enumerate the sentences of L (effectively so, by Theorem 3.1). So take such an enumeration, and imagine God going along the list and removing all the false sentences. That will turn our listing into a pruned enumeration containing only true sentences. ⊠

(b) To state our second corollary, we need a simple definition:

> A set of wffs Σ is *axiomatizable* iff 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).

Then it is immediate that

> **Theorem 5.6** *The set of true sentences of a sufficiently expressive language L is not axiomatizable.*

Proof Suppose otherwise, i.e. suppose that T is an axiomatized formal theory, framed in a sufficiently expressive language L, such that the T-theorems are just the truths expressible in that language. Then, because it is properly axiomatized, T's theorems could be effectively enumerated (by the last part of Theorem 3.1). So the truths of L could be effectively enumerated, contrary to Theorem 5.4. Hence there can be no such theory as T. ⊠

(c) Finally, we just run again the argument given in the preamble to this chapter. So: suppose we build an axiomatized formal theory T in a sufficiently expressive language L. Then because it is axiomatized, T's theorems *can* be effectively enumerated. On the other hand, because T's language is sufficiently expressive, the truths expressible in its language *cannot* be effectively enumerated. There is therefore a mismatch between the truths and the T-theorems here.

Now suppose that T is a *sound* theory, i.e. its theorems are all true. The mismatch between the truths and the T-provable sentences must then be due to there being truths which T can't prove. Suppose φ is one of these. Then T doesn't prove φ; and since $\neg\varphi$ is false, T doesn't prove that either. Hence

> **Theorem 5.7** *If T is a sound axiomatized theory whose language is sufficiently expressive, then T cannot be negation complete.*

Astonishing! We have reached an arithmetical incompleteness theorem already. And note, by the way, that we can't patch T up and make it complete by adding more true axioms and/or a richer truth-preserving logic to get another properly axiomatized theory T'. For T' will still be sound, still have a sufficiently expressive language, and hence still be incomplete. So we could equally well call Theorem 5.7 an *incompletability* theorem.

The great mathematician Paul Erdős had the fantasy of The Book in which God keeps the neatest and most elegant proofs of mathematical theorems. Our proof of Theorem 5.4 and hence of our first incompletability theorem surely belongs in The Book.[6]

[6] For more on Erdős's conceit of proofs from The Book, see Aigner and Zielger (2004).

6 Sufficiently strong arithmetics

Theorem 5.7, our first shot at an incompleteness theorem, applies to sound theories. But we have already remarked in Section 1.2 that Gödel's arguments show that we don't need to assume soundness to prove incompleteness. In this chapter we see how to argue from *consistency* to incompleteness.

But if we are going to weaken one assumption (from soundness to mere consistency) we'll need to strengthen another assumption: we'll now consider theories that don't just *express* enough but which can *capture*, i.e. *prove*, enough.

Starting in Chapter 8, we'll begin examining various 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 different theories are capable of proving. For the moment, however, we are continuing to proceed the other way about. In the previous chapter, we considered theories that have sufficiently expressive languages, and so can express what we'd like any arithmetic to be able to express. Now we introduce the companion concept of a *sufficiently strong* theory, which is one that by definition can prove what we'd like any moderately competent theory of arithmetic to be able to prove about decidable properties of numbers. We then establish some easy but quite deep results about such theories.

6.1 The idea of a 'sufficiently strong' theory

Suppose that P is some effectively decidable property of numbers, i.e. one for which there is a mechanical procedure for deciding, given a natural number n, whether n has property P or not.

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. 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 naturally 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, $T \vdash \neg\varphi(\overline{\mathsf{n}})$. In short, we want T to capture P (in the sense of Section 4.6).

The suggestion therefore is that, if P is any effectively 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* iff it captures all effectively decidable numerical properties.

And it seems a reasonable and desirable condition on a formal theory of the arithmetic of the natural numbers that it be sufficiently strong.[1]

Much later (in Section 30.1), when we've done some more investigation into the general idea of effective decidability, we'll finally be in a position to warrant the claim that some simple, intuitively sound, and (by then) very familiar theories built in L_A do indeed meet this condition. We will 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.

6.2 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 6.1** *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 3.1 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 $T \vdash \neg\varphi_n(\overline{\mathsf{n}})$.

[1] Why is being sufficiently expressive defined in terms of expressing *relations*, and being sufficiently strong defined in terms of capturing *properties*? No deep reason at all. We could have made the definitions symmetric, defining each in terms of expressing/capturing all decidable properties *and* relations. But we've chosen instead to build into the respective definitions only what we actually need to build in if we are to get out our respective theorems.

[2] The undecidability of arithmetic was first shown by Church (1936b). For a neater proof, see Tarski et al. (1953, pp. 46–49). The informal proof as given here is due to Timothy Smiley, who was presenting it in Cambridge lectures in the 1960s. The first published version of it which I know is in Hunter (1971, pp. 224–225).

Note that the construction here links the subscripted index with the standard numeral to be substituted for the variable in $\neg\varphi_n(\mathsf{x})$. So this is a cousin of the 'diagonal' construction which we encountered in Section 2.3 (compare the final comment on the proof of Theorem 2.1).

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 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, in particular, that 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:

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. ⊠

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 the results of boringly mechanical reasoning about decidable properties of individual numbers, it must itself cease to be decidable.

6.3 Another incompleteness theorem

Now let's put together Theorem 3.2, *Any consistent, axiomatized, negation-complete formal theory is decidable*, and Theorem 6.1, *No consistent, sufficiently strong, axiomatized formal theory of arithmetic is decidable.* These, of course, immediately entail

> **Theorem 6.2** *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 $\neg\varphi$ in its language, neither of which is a theorem. But one of the pair 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 sentences of its language which T cannot decide.

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 φ is a true sentence of arithmetic that T can't prove or disprove. The theory T^+ which you get by adding φ as a new axiom to T will, of course, now trivially prove φ, so we've plugged that gap. But note that T^+ is consistent (for if T^+, i.e. $T + \varphi$, were inconsistent, then $T \vdash \neg\varphi$ contrary to hypothesis). And T^+ is sufficiently strong (since it can still prove everything T can prove). It is still decidable which wffs are axioms of T^+, so the theory still counts as a properly axiomatized formal theory. So T^+ is another c.s.s.a. theory and Theorem 6.2 applies: so there is a wff φ^+ (distinct from φ, of course) which is again true-on-interpretation but which T^+ cannot decide (and if T^+ can't prove either φ^+ or $\neg\varphi^+$, then neither can the weaker T). In sum, the c.s.s.a. theory T is therefore not only incomplete but also in a good sense incompletable.[3]

Which is another proof for The Book.

[3]Perhaps we should note that, while the informal incompleteness argument of Chapter 5 depended on assuming that there are general-purpose programming languages in which we can specify (the equivalent of) any numerical algorithm, the argument of this chapter doesn't require that assumption. On the other hand, our previous incompleteness result didn't make play with the idea of theories strong enough to capture all decidable numerical properties, whereas our new incompleteness result does. What we gain on the roundabouts we lose on the swings.

7 Interlude: Taking stock

7.1 Comparing incompleteness arguments

Our informal incompleteness theorems, Theorems 5.7 and 6.2, aren't the same as Gödel's own theorems. But they are cousins, and they seem quite terrific results to arrive at so very quickly.

Or are they? Everything depends, for a start, on whether the ideas of a 'sufficiently expressive' arithmetic language and a 'sufficiently strong' theory of arithmetic are in good order.

Now, as we've already briefly indicated in Section 2.2, there are a number of standard, well-understood, 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 and relations. In fact, these are the properties/relations whose application can be decided by a Turing machine. The specification 'all decidable relations/properties of numbers' can therefore be made perfectly clear. And hence the ideas of a 'sufficiently expressive' language (which expresses all decidable two-place numerical relations) and a 'sufficiently strong' theory (which captures all decidable properties of numbers) can also be made perfectly clear.

But by itself, that observation doesn't take us very far. For it leaves wide open the possibility that a language expressing all decidable relations or a theory that captures all decidable properties has to be very rich indeed. However, we announced right back in Section 1.2 that Gödel's own arguments rule out complete theories even of the truths of basic arithmetic. Hence, if our easy Theorems are to have the full reach of Gödel's work, we'll really have to show the language of basic arithmetic is sufficiently expressive, and that a theory built in that language can be sufficiently strong.

In sum, if something like our argument for Theorem 5.7 is to be used to establish a variant of one of Gödel's own results, then it needs to be augmented with (i) a general treatment of the class of decidable properties/relations, *and* (ii) a proof that (as we claimed) even L_A can express the decidable two-place relations. And if something like our argument for Theorem 6.2 is to be used, it needs to be augmented by (iii) a proof that (as we claimed) common-or-garden theories couched in L_A can be sufficiently strong.

But even with (i), (ii) and (iii) in play, there would still remain a significant difference between our easy theorems and Gödel's arguments. For our lines of argument don't yet give us any specific examples of unprovable truths.[1] By

[1] When we look at the proofs in Chapter 5, we can see that some unprovable wffs will have

contrast, Gödel's proof tells us how to take a consistent theory T and actually construct a true but unprovable-in-T sentence (the one that encodes 'I am unprovable in T'). Moreover, Gödel does this without needing the general treatment in (i) and without needing all of (ii)/(iii) either.

There is a significant gap, then, between our two intriguing, quickly-derived, but informal theorems and the industrial-strength results 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. But before we press on, let's highlight two very important general lessons we can already learn from our informal theorems:

A. Arguments for incompleteness come in (at least) two flavours. First, we can combine the premiss (i) that we are dealing with a *sound* theory with the premiss (ii) that our theory's language is *expressively* rich enough. Or second, we can weaken one assumption and beef up the other: in other words, we can use the weaker premiss (i′) that we are dealing with a *consistent* theory but add the stronger premiss (ii′) that our theory can *prove* enough facts. We'll see that Gödel's arguments too come in these two flavours.

B. Arguments for incompleteness don't have to depend on the construction of Gödel sentences that somehow say of themselves that they are unprovable. Neither of our informal proofs do. (More basically: neither proof depends on Gödel's trick of associating code-numbers with formal expressions.)

7.2 A road-map

So now we turn to Gödel's proofs. And to avoid getting lost in what follows, it will help to have in mind an overall road-map of the route we are taking:

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 8–10)

2. These systems are framed in L_A, the language of basic arithmetic. So they only have 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 not only express but capture all the primitive recursive functions, properties and relations – a major theorem that was, in essence, first proved by Gödel. (Chapters 11–13)

the form $\neg\exists \mathsf{j}\mathsf{R}(\mathsf{j}, \overline{\mathsf{n}})$. But that's not terrifically informative as we don't yet know anything about what R itself has to look like!

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 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 15.)

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. Given PA is sound and only proves truths, G can't be provable; hence G is true; hence ¬G is false, and so is also unprovable in PA. In sum, given PA is sound, it cannot decide G. Further, it turns out that we can drop the *semantic* assumption that PA is sound. Using the fact that PA can capture relations like *Prf* (as well as merely express them), we can still show that G is undecidable while just making a *syntactic* assumption (it is enough that PA is 'ω-consistent'). (Chapter 16)

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 a new true-but-unprovable wff G′. Throwing in G′ as a further axiom then gives us another theory for which we can generate yet another true-but-unprovable wff. And so it goes. PA is therefore not only incomplete but incompletable. In fact, *any* properly axiomatized consistent theory that contains the weak theory Q is incompletable. (Chapter 17)

Just one comment. This summary makes Gödel's formal proofs in terms of *primitive recursive* properties and relations sound rather different from our informal proofs using the idea of theories which express/capture *decidable* properties

or relations. But a link can be made when we note that the primitive recursive properties and relations are in fact a large subclass of the intuitively decidable properties and relations. Moreover, showing that Q and PA can express/capture all primitive recursive properties and relations takes us most of the way to showing that those theories are sufficiently expressive/sufficiently strong.

However, we'll leave exploring this link until much later. Only after we have travelled the original Gödelian route (which doesn't presuppose a *general* account of computability) will we return to consider how to formalize the arguments of Chapters 5 and 6 (a task which *does* presuppose such a general account).

8 Two formalized arithmetics

We now move on from the generalities of the previous chapters, and look at some particular formal arithmetics. In this chapter, we limber up by looking at Baby Arithmetic, and then we start exploring Robinson Arithmetic. Later, in Chapter 10, we'll be introducing Peano Arithmetic, the strongest of our initial range of formal arithmetics.

These theories differ in strength, but they do share one key feature: *the theories' deductive apparatus is no richer than familiar first-order logic.* So we can quantify, perhaps, over all *numbers*: but our theories will lack second-order quantifiers, i.e. we can't quantify over all *numerical properties*.[1]

8.1 BA, Baby Arithmetic

We begin with a *very* simple theory 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 (a) its language, (b) its deductive apparatus, and (c) its 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): hence there is a single individual constant '0', the one-place function symbol 'S', and the two-place function symbols '+' and '×'. So $\mathcal{L}_B$ contains the standard numerals. However, $\mathcal{L}_B$'s logical apparatus is restricted. As we said, it lacks quantifiers and variables. But it has the identity sign (so that it can express equalities), and negation (so that it 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' still 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 terms,[2] then Leibniz's Law will allow us to infer $\varphi(\rho)$ from the premisses $\varphi(\tau)$ and $\tau = \rho$ or $\rho = \tau$.

[1] Since we are treating properties extensionally, quantifying over numerical properties would be tantamount to quantifying over their extensions, i.e. over *sets* of numbers. We say more about second-order arithmetics in Chapter 22.

[2] They will be *closed* terms, in fact, in the sense of Section 4.3.

(c) Now for the (non-logical) 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. 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* (i.e. general templates) 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 $\quad 0 \neq \mathsf{S}\zeta$

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

Let's pause to show that instances of these schemata do indeed determine that different terms in the sequence 0, S0, SS0, 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 $|m - n| - 1 = j \geq 0$). Now, arguing inside BA, assume $\overline{\mathsf{m}} = \overline{\mathsf{n}}$ as a temporary supposition (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). Then 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 $0 = \mathsf{S}\bar{\mathsf{j}}$, or else (ii) we derive $\mathsf{S}\bar{\mathsf{j}} = 0$ and can use the symmetry of identity again to conclude $0 = \mathsf{S}\bar{\mathsf{j}}$. That shows that $\mathsf{BA} \vdash \overline{\mathsf{m}} = \overline{\mathsf{n}} \rightarrow 0 = \mathsf{S}\bar{\mathsf{j}}$. But $\mathsf{BA} \vdash 0 \neq \mathsf{S}\bar{\mathsf{j}}$ (by an instance of Schema 1). Hence, $\mathsf{BA} \vdash \overline{\mathsf{m}} \neq \overline{\mathsf{n}}$. ⊠

We next 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 $\quad \zeta + 0 = \zeta$

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 '0' define 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 'S0' to define how to add two ($SS0$) in terms of adding $S0$. We can then invoke the same Schema again to define 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 an axiom too:

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

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

Instances of Schema 5 tell us the result of multiplying by zero. Instances of Schema 6 with 'ξ' replaced by '0' define 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 (multiply by one and do 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 a properly axiomatized theory.

8.2 BA is negation complete

We start by noting the easy result that BA's axioms can be used to derive all the correct results about the simple addition or multiplication of two numbers.

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

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

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

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

Which gives us what we want:

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

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

What about more complex cases? Let's say that an *equation* of BA is a wff of the form $\tau = \rho$, where τ and ρ are closed terms (of any complexity). 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 form $\bar{\mathsf{j}} + \bar{\mathsf{k}} = \overline{\mathsf{m}}$ or $\bar{\mathsf{j}} \times \bar{\mathsf{k}} = \bar{\mathsf{n}}$. And given a more complex closed term τ, involving nested additions, multiplications and applications of the successor function, we can 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 identities of the form $\bar{\mathsf{j}} + \bar{\mathsf{k}} = \overline{\mathsf{m}}$ and $\bar{\mathsf{j}} \times \bar{\mathsf{k}} = \bar{\mathsf{n}}$, thereby evaluating the inner-most bracketed expressions. Next substitute these results into the logical truth $\tau = \tau$ using Leibniz's Law, and that will enable us to derive

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

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

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

And we are done, as the expression on the right *is* a numeral.

Further, it is evident 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.

Hence, 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 $\tau = \overline{\mathsf{m}}$, $\rho = \bar{\mathsf{n}}$. But we've already shown in the previous section that if $m \neq n$, BA proves $\overline{\mathsf{m}} \neq \bar{\mathsf{n}}$. So, if $m \neq n$, a BA proof of $\tau \neq \rho$ follows using Leibniz's Law twice. ⊠

These results (1) and (2) in turn imply:

Theorem 8.1 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 equation and (2) prove the negation of each false equation. So, by a theorem of propositional logic, we can derive any true truth-functional combination of atoms (equations), i.e. prove any true sentence; likewise, we can also derive the negation of any 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 3.2. 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.

8.3 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). Our theory'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 formal axioms governing particular numbers) by a single generalized Axiom. For example, we can replace the first two Schemata governing the successor function by

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

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

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

Note, however, that while these Axioms tell us that zero isn't a successor, they leave it open whether 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 explicitly rule them out:

Axiom 3 $\quad \forall x(x \neq 0 \rightarrow \exists y(x = Sy))$

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

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

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.[3]

8.4 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 $\mathcal{L}_A$-wffs, where $\mathcal{I}_D$ still makes Q's Axioms true but allows cases where 'adding' a 'number' to the 'zero' yields a different 'number'. 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 could be e.g. Kurt Gödel and his friend Albert Einstein). 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 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 makes χ false.

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

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. We can leave it as an exercise to fill in suitable details. Then, with the details filled in, we will have an overall interpretation which makes the axioms of Q true and χ false. So Q can't prove χ. ⊠

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. Hence, in sum, $\mathsf{Q} \nvdash \chi$ and $\mathsf{Q} \nvdash \neg\chi$.[4] Which gives us the utterly unsurprising

Theorem 8.2 Q *is not negation complete.*

Of course, we've already announced that Gödel's incompleteness theorem is going to prove that *no* sound 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.

8.5 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 very pretty properties. In particular, and perhaps rather surprisingly, we'll later be able to show that Q *is sufficiently strong in the sense of Chapter 6.* For 'sufficient strength' is a matter of being able to *case-by-case* prove enough wffs about decidable properties of individual numbers. And Q's hopeless weakness at proving generalizations doesn't stop it doing that.

So that's why Q is interesting. Suppose a theory of arithmetic is formally axiomatized, consistent and can prove everything Q can prove (those do seem very modest requirements). Then what we've just announced and promised to prove is that any such theory will be sufficiently strong. And therefore Theorem 6.2 will apply – any such theory will be incomplete.

However, establishing the crucial claim that Q *does* have sufficient strength to capture all decidable properties has to be business for much later – plainly, we can only establish it when we have a quite general theory of decidability to hand.

What we *will* prove quite soon is a rather weaker claim about Q: in Chapter 13, we show that it can capture all 'primitive recursive' properties, where these form an important subclass of the decidable properties. This major theorem will be a crucial load-bearing part of our proofs of various Gödel-style incompleteness theorems. The following chapter goes through some necessary preliminaries.

[4]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.

9 What Q can prove

As we saw, Robinson's Q is a very weak theory of arithmetic: but we'll explore here what it *can* establish. Unavoidably, some of the detailed *proofs* of our claims do get boringly fiddly; so you are very welcome to skip as many of them as you like (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. So here's a quick guide through the first eight sections of this busy chapter.

1. We begin with some remarks about different systems of first-order logic.

2. We 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 4.5, (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 note that Q can formally prove a range of expected results about the less-than-or-equal-to relation. (But we relegate detailed proofs to a ninth section at the end of the chapter.)

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 class of 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.

7. We next show that Q knows enough about bounded quantifiers to be able to correctly decide every *bounded* sentence – i.e. prove a Δ_0 sentence if it is true, disprove it if it is false. And that quickly gives us the key theorem of this chapter: Q *can also prove any true existentially quantified bounded wff* – it can prove any true Σ_1 sentence, i.e. is 'Σ_1-complete'.

8. We then prove a couple of intriguing corollaries of that last result.

9.1 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 the same sentences as conclusions from given sentences as premisses. Let's contrast, in particular, Hilbert-style axiomatic systems with natural deduction systems.

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 – perhaps with some restrictions – that 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 simply 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, it will 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 lay 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 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 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 proof 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 Hilbert-style system, see e.g. 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 Fitch (1952).

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

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

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 $m = 2$, $n = 1$.

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; we drop unnecessary brackets):[3]

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!	∃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 the Existential Elimination rule: if the temporary supposition $\varphi(a)$ leads to contradiction, for arbitrary a, then $\exists v\varphi(v)$ must lead to contradiction.

And having done the proof for the case $m = 2$, $n = 1$, inspection reveals that we can use the same general pattern of argument to show $Q \vdash \neg\exists v(v + \overline{m} = \overline{n})$ whenever $m > n$. (Exercise: convince yourself that this claim is true!) So we are done. ⊠

[3] Does it need saying? 'LL' again indicates the use of Leibniz's Law, 'MP' stands for modus ponens, 'RAA' for reductio ad absurdum.

9.3 Adding '≤' to Q

Given the result we've just proved, we can sensibly add the standard symbol '≤' to L_A, the language of Q, defined so that whatever we put for 'ξ' and 'ζ', $\xi \leq \zeta$ is short for $\exists \mathsf{v}(\mathsf{v} + \xi = \zeta)$.[4] Since it so greatly helps readability, we'll henceforth make very free use of this abbreviatory symbol inside formal arithmetics.

We will also adopt a second, closely related, convention. In informal mathematics we often want to say that all/some numbers less than or equal to a given number have some particular property. We can now express such claims in formal arithmetics by wffs of the shape $\forall \xi(\xi \leq \kappa \rightarrow \varphi(\xi))$ and $\exists \xi(\xi \leq \kappa \wedge \varphi(\xi))$, where '≤' is to be unpacked as we've just explained. And it is standard to further abbreviate such wffs with *bounded quantifiers* by $(\forall \xi \leq \kappa)\varphi(\xi)$ and $(\exists \xi \leq \kappa)\varphi(\xi)$ respectively.

9.4 Q is order-adequate

Q can case-by-case capture the less-than-or-equal-to relation. We'll now remark that Q can also prove a bunch of general facts about this relation. Let's say, for brevity, that a theory T is *order-adequate* if the following nine propositions hold:

O1. $T \vdash \forall \mathsf{x}(0 \leq \mathsf{x})$.

O2. For any n, $T \vdash \forall \mathsf{x}(\{\mathsf{x} = 0 \vee \mathsf{x} = 1 \vee \ldots \vee \mathsf{x} = \overline{\mathsf{n}}\} \rightarrow \mathsf{x} \leq \overline{\mathsf{n}})$.

O3. For any n, $T \vdash \forall \mathsf{x}(\mathsf{x} \leq \overline{\mathsf{n}} \rightarrow \{\mathsf{x} = 0 \vee \mathsf{x} = 1 \vee \ldots \vee \mathsf{x} = \overline{\mathsf{n}}\})$.

O4. For any n, if $T \vdash \varphi(0)$, $T \vdash \varphi(1)$, …, $T \vdash \varphi(\overline{\mathsf{n}})$, then $T \vdash (\forall \mathsf{x} \leq \overline{\mathsf{n}})\varphi(\mathsf{x})$.

O5. For any n, if $T \vdash \varphi(0)$, or $T \vdash \varphi(1)$, …, or $T \vdash \varphi(\overline{\mathsf{n}})$, then $T \vdash (\exists \mathsf{x} \leq \overline{\mathsf{n}})\varphi(\mathsf{x})$.

O6. For any n, $T \vdash \forall \mathsf{x}(\mathsf{x} \leq \overline{\mathsf{n}} \rightarrow \mathsf{x} \leq \mathsf{S}\overline{\mathsf{n}})$.

O7. For any n, $T \vdash \forall \mathsf{x}(\overline{\mathsf{n}} \leq \mathsf{x} \rightarrow (\overline{\mathsf{n}} = \mathsf{x} \vee \mathsf{S}\overline{\mathsf{n}} \leq \mathsf{x}))$.

O8. For any n, $T \vdash \forall \mathsf{x}(\mathsf{x} \leq \overline{\mathsf{n}} \vee \overline{\mathsf{n}} \leq \mathsf{x})$.

O9. For any $n > 0$, $T \vdash (\forall \mathsf{x} \leq \overline{\mathsf{n-1}})\varphi(\mathsf{x}) \rightarrow (\forall \mathsf{x} \leq \overline{\mathsf{n}})(\mathsf{x} \neq \overline{\mathsf{n}} \rightarrow \varphi(\mathsf{x}))$.

(Obviously, $\overline{\mathsf{n-1}}$ is the numeral for $n-1$.) Then we have the following summary result:

[4] 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 '≤' 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 '≤' as our basic symbol rather than '<', which could have been defined by $\xi < \zeta =_{\text{def}} \exists \mathsf{v}(\mathsf{Sv} + \xi = \zeta)$.

Theorem 9.1 Q *is order-adequate.*

This theorem is absolutely pivotal for what follows. It does, however, belong very squarely to the class of 'trivial but tiresome' results. It is fiddly to check that Q satisfies the nine conditions: but none of the proofs involves anything particularly interesting – we'll give a few examples in the final section of this chapter. If you have a taste for logical brain-teasers, then by all means see how many of the nine conditions you can verify. Otherwise you are very welcome to skip the proofs and take the results on trust.[5]

9.5 Defining the Δ_0, Σ_1 and Π_1 wffs

(a) Why bother establishing that Q is order-adequate? Because, as we'll see, this implies 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 see 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 show 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* existential *quantifiers followed up by a* bounded *kernel wff.*

And a bounded wff – a Δ_0 wff, to use standard jargon – is one which 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 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. But, over the next three sections, we had better say the same thing again much more slowly, filling in the details as we go along. We start with the key definitions:

[5] '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.

[6] 'Hold on! What exactly is meant by "bounded" quantification here? After all, a bounded quantification like $(\exists x \leq 2)Fx$ is just short for $\exists x(x \leq 2 \wedge Fx)$, which involves a perfectly *ordinary* quantifier, running over all the domain. So, when abbreviations are unpacked, all quantifiers are on a par.' In a sense, that's 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.

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. Now for the full class of Δ_0 wffs:

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 iff 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 iff it is logically equivalent to a strictly Σ_1 wff.

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

(c) Let's pause for a comment on notation. It is worth remarking that 'Σ' in the standard label 'Σ_1' comes from an old alternative symbol for the existential quantifier, as in $\Sigma x Fx$ – that's a Greek 'S' for '(logical) sum'. Likewise the 'Π' in 'Π_1' comes from corresponding symbol for the universal quantifier, as in $\Pi x Fx$ – that's a Greek 'P' for '(logical) 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 Π_2 wff is one that starts with *two* blocks of quantifiers, a block of universal quantifiers followed by a block of existential quantifiers followed by a bounded kernel. Similarly, a Σ_0 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 equally current.[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 minor ways across different presentations. Don't be fazed by this. 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 mostly doesn't matter if we

[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 the distinctness requirement? Because $(\forall \mathsf{x} \leq \mathsf{x})\varphi(\mathsf{x})$, for example, unpacks as $\forall \mathsf{x}(\mathsf{x} \leq \mathsf{x} \rightarrow \varphi(\mathsf{x}))$ which is equivalent to the *unbounded* $\forall \mathsf{x}\varphi(\mathsf{x})$. Not what we want!

[9] The now conventional Σ_n/Π_n notation goes back to the late 1950s: but the original recognition of the importance of the hierarchy of formulae with increasing quantifier complexity is due to Kleene (1943).

alternatively define a strictly Σ_1 wff as one which starts with a *single* existential quantifier before its Δ_0 kernel (for a proof, see Section 10.3).

(d) A few examples might help. Allowing abbreviations,

1. $1 \leq 3$, $y + 1 = x$, and $SS(y \times 1) = y \times SSSy$ are atomic Δ_0 wffs.
2. $y + 1 = x$, $(\exists x \leq 2)\, x \neq 1$, $(\exists y \leq 3)\, x = 1 + y$, $\neg(\exists z \leq 5)(\forall y \leq x)\, y \leq z + x$, and $(1 \leq x \rightarrow (\exists y \leq x)Sy = x)$ are Δ_0 wffs.
3. $\exists x \exists y\, y + 1 = x$ and $\exists x(\exists y \leq x)\, y + 3 = Sx$ are both strictly Σ_1 wffs. And $\forall x \forall y\, x + y = y + x$ and $\forall x(1 \leq x \rightarrow (\exists y \leq x)Sy = x)$ are strictly Π_1 wffs.
4. $\neg\forall x \forall y\, y + 1 \neq x$ and $\neg\forall x(\forall y \leq x)\, y + 3 \neq Sx$ are Σ_1 wffs since they are logically equivalent to our sample strictly Σ_1 wffs.

9.6 Some easy results

(a) Here are 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 $(\varphi \wedge z = z)$ is also Δ_0. So $\exists z(\varphi \wedge z = z)$ is strictly Σ_1 and $\forall z(\varphi \wedge z = z)$ is strictly Π_1. But those two wffs are each logically equivalent to the original φ; hence φ also counts as both Σ_1 and Π_1. ⊠

Note that the last argument shows that being Σ_1 and being Π_1 are *not* exclusive – though, of course, being *strictly* Σ_1 trivially excludes being *strictly* Π_1.

(b) Another basic result:

4. We can work out the truth or falsity of any closed Δ_0 wff by an algorithmic calculation.

This holds essentially because we only have to look through a finite number of cases in order to deal with any *bounded* quantifications. And you might think our claim is too obvious really to need a proof. Fair enough. But we'll give an argument all the same (skip if you must!) – for it does nicely illustrate a standard technique, namely proving a result about all wffs in some class by an *induction on the complexity of the wff*.[10]

[10] We should perhaps stress here that we are not going to be using an inductive argument *inside* Q; we can't do that because Q lacks induction axioms (compare PA, introduced in the next chapter). Rather, we are standing outside Q and using everyday informal inductive reasoning to show something *about* Q. If you haven't met this kind of argument before, you might find the following helpful: Boolos et al. (2002, pp. 109–110) or Leary (2000, pp. 16–19).

Proof 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.

Now (α), the degree 0 sentences are the *atomic* Δ_0 sentences; and we can trivially calculate the truth-value of any such sentence.

And (β), suppose that we can algorithmically 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 $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \rightarrow \psi)$ or $(\varphi \leftrightarrow \psi)$, where φ and ψ are Δ_0 sentences of degree no greater than k. The truth-value of the relevant φ and ψ is by hypothesis calculable, and hence (using truth-tables) so is the truth value of χ.

ii. χ is of the form $(\forall\xi \leq \overline{\mathsf{n}})\varphi(\xi)$, where $\varphi(\xi)$ is a Δ_0 wff of degree k.[11] Then χ is equivalent to a conjunction $\varphi(\mathsf{0}) \wedge \varphi(\mathsf{1}) \wedge \ldots \wedge \varphi(\overline{\mathsf{n}})$. Each conjunct is therefore a closed Δ_0 wff of degree k, i.e. it is a sentence whose truth-value is by hypothesis calculable. 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.

Putting (α) and (β) together, we can mechanically settle the truth-value of Δ_0 sentences of degree 0 and, if we can settle the truth-values of Δ_0 sentences of degree up to k, we can settle the values of sentences of degree up to $k+1$. Hence, since we *can* settle the truth-values of Δ_0 sentences of degree 0, we can settle the values of Δ_0 sentences of degree 1, and so in turn settle the degree 2 cases, and so on upwards. In sum, by an informal induction on k, we can algorithmically determine the truth-value of *any* Δ_0 sentence, however complex. ⊠

9.7 Q is Σ_1-complete

Suppose Γ stands in for some type of L_A wff (e.g. the Δ_0 wffs or the Σ_1 wffs, etc.), and let T be a theory which includes some arithmetic. Then we'll say

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

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

(Here, 'true' means, of course, true when interpreted as a sentence of arithmetic.) And with that jargon to hand, we can state the following theorem, the key result of this chapter:

Theorem 9.2 Q *is* Σ_1*-complete.*

We can demonstrate this by proving in turn that

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

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

We already know that Q can correctly decide every quantifier-free sentence, i.e. every sentence it shares with BA (see the opening remark of Section 8.4). So (2) extends that simple result to cover sentences with bounded quantifiers. And (3) follows trivially from (2). You can skip the proofs again.

Proof for (1) An atomic Δ_0 *sentence* – i.e. a Δ_0 wff without free variables – is either (i) an equation $\tau_1 = \tau_2$ or else (ii) a wff of the form $\tau_1 \leq \tau_2$ (where τ_1 and τ_2 are closed terms, denoting the numbers t_1 and t_2 respectively). In case (i), we are done, because we already know that Q correctly decides every equation. In case (ii), 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. As before, there are three cases to consider.

i. χ is built using a propositional connective from φ and perhaps 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 (O4) of Section 9.4, Q also proves $(\forall\xi \leq \overline{\mathsf{n}})\varphi(\xi)$. 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 (O5), Q proves $(\exists\xi \leq \overline{\mathsf{n}})\neg\varphi(\xi)$, which easily entails $\neg(\forall\xi \leq \overline{\mathsf{n}})\varphi(\xi)$. 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 (ii).

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 m, n, 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 $\exists \mathsf{x} \exists \mathsf{y} \varphi(\mathsf{x}, \mathsf{y})$, by existential introduction. Evidently the argument generalizes for any number of initial quantifiers, which shows that Q proves all true strictly Σ_1 sentences. So it will prove their true logical equivalents too. ⊠

9.8 Intriguing corollaries

That wasn't very exciting, but the work needed to be done: the theorem that Q – and therefore any stronger arithmetic – is Σ_1-complete will turn out be a key ingredient in various Gödelian incompleteness arguments. But that's for later. For now, let's just note a couple of immediate corollaries of our theorem.

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.[12] As we've noted before, no proof of the conjecture is currently known.

Now suppose that Goldbach's conjecture is *false*. Then its *negation* will be a true Σ_1 sentence, and hence – by Theorem 9.2 – be provable in Q. While if Goldbach's conjecture is *true*, then its negation will be false and therefore not provable in Q (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. Equivalently, it is enough to decide whether the conjecture is consistent with Q.

The argument evidently generalizes to any Π_1 arithmetical claim: Fermat's Last Theorem is another example. Hence,

> **Theorem 9.3** *A Π_1 sentence φ is true iff $\neg\varphi$ is not logically deducible from* Q*, i.e. iff φ is consistent with* Q*.*

So, if we had a method which we could use to decide whether a wff φ 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 30.4).

Here's a further development. We need two final definitions:

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

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

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

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

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

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

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

i. An interpreted language L_2 *includes* the language L_1 iff (a) every L_1-wff is also a wff of L_2, perhaps allowing for some definitional extensions of L_2, and (b) the copies of L_1-wffs in L_2 have the same truth-conditions as before, i.e. the copies are true just when their originals are true.

ii. A theory T_2 *extends* T_1 if T_2's language includes T_1's language and further, T_2 can prove all T_1's theorems (and perhaps more).

Then it is immediate that, since Q is Σ_1-complete, so is any theory T which extends Q. It then follows that

Theorem 9.4 *If T extends* Q*, T is consistent iff it is Π_1-sound.*

Proof T is Π_1-sound iff every Π_1 sentence that T proves is true as a sentence of arithmetic. 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, so is Π_1-sound.

The converse is trivial, since if T is inconsistent, we can derive anything in T, including false Π_1 sentences and so T isn't Π_1-sound. ⊠

This is, in its way, a rather remarkable observation. It means that we don't have to fully *believe* a theory T – i.e. don't have to accept *all* its theorems are true on the interpretation built into T's language – in order to use it to establish that some Π_1 arithmetic generalization is true. We just have to believe that T is a consistent theory which extends Q.

Another observation. Suppose Q_G is the theory you get by adding Goldbach's conjecture as an additional axiom to Q. Then, by Theorem 9.4, Q_G is consistent only if the conjecture is true, since the conjecture is a Π_1 theorem of Q_G. But no one knows whether Goldbach's conjecture *is* true; so no one knows whether Q_G is consistent. Which is a powerful reminder that even *very* simple theories may not wear their consistency on their face.

9.9 Proving Q is order-adequate

Finally in this chapter, we'll (partially) verify Theorem 9.1 since so much depends on it. But you must not get bogged down in the tiresome details: so feel free to skip this section!

If you are still reading, then let's first quickly clarify notation. We are using '$\overline{\mathsf{n}}$' to indicate L_A's standard numeral for n. So '$\overline{\mathsf{n}} + 1$' 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. (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 (O1) For arbitrary a, Q proves $\mathsf{a} + \mathsf{0} = \mathsf{a}$, hence $\exists \mathsf{v}(\mathsf{v} + \mathsf{0} = \mathsf{a})$, i.e. $\mathsf{0} \leq \mathsf{a}$. Generalize to get the desired result. ⊠

Proof for (O2) Arguing inside Q, suppose that $\mathsf{a} = \mathsf{0} \vee \mathsf{a} = \mathsf{1} \vee \ldots \vee \mathsf{a} = \overline{\mathsf{n}}$. We showed in Section 9.2 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 $(\mathsf{a} = \mathsf{0} \vee \mathsf{a} = \mathsf{1} \vee \ldots \vee \mathsf{a} = \overline{\mathsf{n}}) \rightarrow \mathsf{a} \leq \overline{\mathsf{n}}$. The desired result is immediate since a was arbitrary. ⊠

Proof for (O3) This is trickier: we'll argue by an informal induction. Suppose we can show that (α) the target wff is provable for $n = 0$. And suppose we can also show that (β) if it is provable for $n = k$ it is provable for $n = k+1$. Together, these establish the desired conclusion that the target wff is provable for all n. Hence it is enough to show that (α) and (β) both hold.[13]

(α) To show that the target wff is provable for $n = 0$, we need a proof of $\forall \mathsf{x}(\mathsf{x} \leq \mathsf{0} \rightarrow \mathsf{x} = \mathsf{0})$. It is enough to suppose, inside a Q proof, that $\mathsf{a} \leq \mathsf{0}$, for arbitrary a, and deduce $\mathsf{a} = \mathsf{0}$. So suppose $\mathsf{a} \leq \mathsf{0}$, i.e. $\exists \mathsf{v}(\mathsf{v} + \mathsf{a} = \mathsf{0})$. Then for some b, $\mathsf{b} + \mathsf{a} = \mathsf{0}$. Now Axiom 3 is equivalent to $\forall \mathsf{x}(\mathsf{x} = \mathsf{0} \vee \exists \mathsf{y}(\mathsf{x} = \mathsf{Sy}))$, so it tells us that either (i) $\mathsf{a} = \mathsf{0}$, or (ii) $\mathsf{a} = \mathsf{Sa'}$ for some $\mathsf{a'}$. But (ii) implies $\mathsf{b} + \mathsf{Sa'} = \mathsf{0}$, so by Axiom 5 $\mathsf{S}(\mathsf{b} + \mathsf{a'}) = \mathsf{0}$, contradicting Axiom 1. That rules out case (ii). Therefore $\mathsf{a} = \mathsf{0}$. Which establishes (α).

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

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

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

Hence, since either (i) or (ii) holds, $\mathsf{a} = \mathsf{0} \vee \mathsf{a} = \mathsf{1} \vee \mathsf{a} = \mathsf{2} \vee \ldots \vee \mathsf{a} = \overline{\mathsf{k+1}}$. Which establishes ($\beta$). ⊠

Proofs for (O4) to (O7) Exercises!

Proof for (O8) We show that (α) the target wff is provable in Q for $n = 0$, and that (β) 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.

(α) We saw in proving (O1) that for any a, $\mathsf{0} \leq \mathsf{a}$. A fortiori, $\mathsf{0} \leq \mathsf{a} \vee \mathsf{a} \leq \mathsf{0}$. Generalizing gives us the desired result for $n = 0$.

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

[13] A notational warning for any reader casually browsing through: in what follows, the prime symbol as in '$\mathsf{a'}$' does *not* indicate the successor function! – cf. Section 4.3, fn. 1.

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 (O6)
iii.	$\overline{\mathsf{k}} \leq \mathsf{a} \rightarrow (\overline{\mathsf{k}} = \mathsf{a} \ \vee \ \overline{\mathsf{k}+1} \leq \mathsf{a})$	By (O7)

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 (O9) Another exercise!

10 First-order Peano Arithmetic

Q is Σ_1-complete, a fact which will turn out to be very important. But, as we saw, in other ways Q is an extremely weak theory. To derive elementary general truths like $\forall x(0 + x = x)$ that are beyond Q's reach, we obviously will have to use a formal arithmetic that incorporates some stronger axiom(s) for proving quantified wffs. This chapter explains the *induction axioms* we need to add, working up to the key theory PA, *first-order Peano Arithmetic*.

10.1 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.

In fact, we used informal inductions in the last chapter. For example, to prove that Q correctly decides all Σ_1 wffs, we in effect began: let n have the property P if Q-correctly-decides-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 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) Our intuitive principle is a generalization covering *any* property of numbers. Hence to frame a corresponding formal version, it seems that we should ideally use a language that enables us to generalize over numerical properties. However, as we announced at the very beginning of Chapter 8, we are currently concentrating on formal theories built in languages like L_A whose logical apparatus involves only 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. But then how can we handle induction?

Our only option is to use a schema again – i.e. to give a template, and say that any wff which fits the template is an induction axiom.[1] So we'll say – to start with – that *any sentence 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 'S' for 'x'. We'll think about what counts as a 'suitable' wff in just a moment: but basically we want expressions $\varphi(x)$ which express genuine properties, i.e. properties which really do fall under the intuitive induction principle.

10.2 Induction and relations

But even before we start worrying about what 'suitable' means, there's more to be said about induction. For we can also use inductive arguments to prove general results about *relations* as well as about monadic properties.

Take a simple illustrative example. Suppose 'Rxyz' abbreviates some open L_A wff with three free variables, which expresses some genuine 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 nice monadic property – the property of standing in the R relation to a and b. The intuitive induction principle should therefore now apply to this property too. Hence the following will be true:

$$(\{R0ab \wedge \forall x(Rxab \rightarrow RSxab)\} \rightarrow \forall xRxab)$$

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

$$\forall y\forall z(\{R0yz \wedge \forall x(Rxyz \rightarrow RSxyz)\} \rightarrow \forall xRxyz)$$

and this proposition is still intuitively warranted. Now, this *isn't* an instance of our Induction Schema. But it *is* the *universal closure* of the instance with $\varphi(x) = Rxyz$ – where the universal closure of a wff is the result of prefixing it with enough universal quantifiers to bind its free variables.

What goes for Rxyz evidently goes in other cases. So we can generalize. *Take any suitable $\varphi(x)$ which has 'x' free and perhaps other variables free too: then the universal closure of the corresponding instance of the Induction Schema is warranted as an axiom.*

[1] 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*.

10.3 Arguing using induction

(a) Which wffs φ, however, are 'suitable' for appearing in the induction schema? We said: ones which express genuine numerical properties (and relations) that fall under the intuitive induction principle. But then the question just becomes: which wffs express genuine properties and relations?

Well, let's start *very* modestly. Suppose $\varphi(\mathsf{x})$ has just 'x' free and has at most bounded quantifiers: in other words, suppose $\varphi(\mathsf{x})$ 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 9.6 (b). Likewise, a Δ_0 wff with two or more free variables surely expresses an entirely determinate relation. So in this section, we'll consider the quite uncontroversial use of induction for Δ_0 wffs.

The following general claims about the successor function, addition and ordering are all provable using Q's Axioms plus induction for Δ_0 wffs:

1. $\forall \mathsf{x}(\mathsf{x} \neq \mathsf{Sx})$
2. $\forall \mathsf{x}(\mathsf{0} + \mathsf{x} = \mathsf{x})$
3. $\forall \mathsf{x} \forall \mathsf{y}(\mathsf{Sx} + \mathsf{y} = \mathsf{S}(\mathsf{x} + \mathsf{y}))$
4. $\forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} + \mathsf{y} = \mathsf{y} + \mathsf{x})$
5. $\forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}(\mathsf{x} + (\mathsf{y} + \mathsf{z}) = (\mathsf{x} + \mathsf{y}) + \mathsf{z})$
6. $\forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}(\mathsf{x} + \mathsf{y} = \mathsf{x} + \mathsf{z} \rightarrow \mathsf{y} = \mathsf{z})$
7. $\forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} \leq \mathsf{y} \vee \mathsf{y} \leq \mathsf{x})$
8. $\forall \mathsf{x} \forall \mathsf{y}((\mathsf{x} \leq \mathsf{y} \wedge \mathsf{y} \leq \mathsf{x}) \rightarrow \mathsf{x} = \mathsf{y})$
9. $\forall \mathsf{x} \forall \mathsf{y} \forall \mathsf{z}((\mathsf{x} \leq \mathsf{y} \wedge \mathsf{y} \leq \mathsf{z}) \rightarrow \mathsf{x} \leq \mathsf{z})$

We'll leave giving a full derivation of each of these wffs as a rather unexciting exercise for those who have a taste for such things. Here are just a few hints and outlines.

Proof sketch for (1) Take $\varphi(\mathsf{x})$ to be $\mathsf{x} \neq \mathsf{Sx}$. Then Q proves $\varphi(\mathsf{0})$ because that's Axiom 1, and proves $\forall \mathsf{x}(\varphi(\mathsf{x}) \rightarrow \varphi(\mathsf{Sx}))$ by contraposing Axiom 2. And then an induction axiom tells us that if we have both $\varphi(\mathsf{0})$ and $\forall \mathsf{x}(\varphi(\mathsf{x}) \rightarrow \varphi(\mathsf{Sx}))$ we can deduce $\forall \mathsf{x}\varphi(\mathsf{x})$, i.e. no number is a self-successor. ⊠

Recall that our deviant interpretation which makes the axioms of Q true while making $\forall \mathsf{x}(\mathsf{0} + \mathsf{x} = \mathsf{x})$ false had Kurt Gödel himself as a self-successor (see Section 8.4). A smidgin of induction, however, rules out self-successors.[2]

[2]Don't be lulled into a false sense of security, though! While induction axioms may rule out deviant interpretations based on self-successors, they don't rule out some other deviant interpretations. See Kaye (1991) for an exploration of 'non-standard models'.

Proof sketch for (2) We showed in Section 8.4 that the wff $\forall x(0 + x = x)$ is *not* provable in Q. But we can derive it once we have induction on the scene.

For take $\varphi(x)$ to be $(0 + x = x)$, which is of course Δ_0. Q implies $\varphi(0)$, since that's just an instance of Axiom 4.

Next we show Q entails $\forall x(\varphi(x) \rightarrow \varphi(Sx))$. Arguing inside Q, it is enough to suppose $\varphi(a)$ and derive $\varphi(Sa)$. So suppose $\varphi(a)$, i.e. $0 + a = a$. But Axiom 5 entails $0 + Sa = S(0 + a)$. So by Leibniz's Law, $0 + Sa = Sa$, i.e. $\varphi(Sa)$.

And now an instance of the Induction Schema tells us that if we have both $\varphi(0)$ and $\forall x(\varphi(x) \rightarrow \varphi(Sx))$ we can deduce $\forall x\varphi(x)$, i.e. $\forall x(0 + x = x)$. ⊠

Proof sketch for (4) Take the universal closure of the instance of the Induction Schema for $\varphi(x) = x + y = y + x$. Then instantiate with an arbitrary parameter to get:

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

To proceed, we can obviously derive $0 + a = a + 0$, the first conjunct in the curly brackets, by using (2) plus Q's Axiom 4.

Now suppose $b + a = a + b$. Then $Sb + a = S(b + a) = S(a + b) = a + Sb$, by appeal to the unproved Result (3), our supposition, and Axiom 5 in turn. So by Conditional Proof $b + a = a + b \rightarrow Sb + a = a + Sb$. Generalizing gets us the second conjunct in curly brackets.

Detach the consequent and then generalize again to get what we want. ⊠

Proof sketch for (7) 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 (O1) of Section 9.9.

Next we show that if we suppose $a \leq b \vee b \leq a$, we can derive $a \leq Sb \vee Sb \leq a$ for arbitrary b, which is enough to 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 (3), $Sc + a = Sb$, 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 $c = Sc'$ for some c'. In the first case $0 + b = a$, so by Result (2) above $a = b$, from which it is trivial that $a \leq Sb$. In the second case, $Sc' + b = a$, and using Result (3) and Axiom 5 we get $c' + Sb = a$, and hence $Sb \leq a$. So $(a \leq Sb \vee Sb \leq a)$ again.

So we can infer $\forall x(a \leq x \vee x \leq a)$. Generalizing gives us the desired result. ⊠

Which all goes to show that proofs by induction do soon get very, *very*, tiresome. But carrying on in the same way, we can in fact prove many of the most obvious general properties of addition and of multiplication too, using only instances of the Induction Schema where φ is a Δ_0 wff.

(b) Let's pause to mention one more result, which is just about worth highlighting as

> **Theorem 10.1** *In any theory which extends* Q *and has induction for* Δ_0 *wff, a* Σ_1 *wff starting with* $n > 1$ *unbounded existential quantifiers is provably equivalent to a* Σ_1 *wff starting with just a* single *unbounded quantifier.*

There are two reasons why this is worth proving. First, we'll actually make use of this lemma later, in Section 19.4. But second, and more immediately, it explains away a common variation you'll find among definitions of the class of (strictly) Σ_1 wffs. Our preferred official definition allows a (strictly) Σ_1 wff to begin with a whole *block* of unbounded existential quantifiers; but it is equally common to insist on there being only *one* unbounded quantifier. Our little result here shows that, in most cases (i.e. in the context of any theory with just a smidgin of induction), this is a distinction which doesn't make a difference.

Proof Observe that the sentence

i. $\exists \mathsf{x} \exists \mathsf{y} \varphi(\mathsf{x}, \mathsf{y})$

is intuitively true in exactly the cases when

ii. $\exists \mathsf{w} (\exists \mathsf{x} \leq \mathsf{w})(\exists \mathsf{y} \leq \mathsf{w}) \varphi(\mathsf{x}, \mathsf{y})$

is true. And first we'll show that (i) and (ii) are indeed provably equivalent, for any φ, using Q plus Δ_0 induction.

Assume (i) and reason inside our formal theory. So we suppose that for some a, b, $\varphi(\mathsf{a}, \mathsf{b})$. But we know from Result (7) above that $\mathsf{a} \leq \mathsf{b} \vee \mathsf{b} \leq \mathsf{a}$. Assume $\mathsf{a} \leq \mathsf{b}$: then we can derive $\mathsf{a} \leq \mathsf{b} \wedge \mathsf{b} \leq \mathsf{b} \wedge \varphi(\mathsf{a}, \mathsf{b})$, which in turn implies $(\exists \mathsf{x} \leq \mathsf{b})(\exists \mathsf{y} \leq \mathsf{b}) \varphi(\mathsf{x}, \mathsf{y})$, and hence (ii). Assume $\mathsf{b} \leq \mathsf{a}$: we can similarly derive (ii). So, arguing by cases, (i) implies (ii).

For the converse implication, unpacking the abbreviations involved in the bounded quantifiers is enough to show that (ii) logically entails (i) and so certainly (ii) implies (i) inside Q.

Now suppose, in particular, that the kernel φ in our example is in fact Δ_0. Then the Σ_1 sentence (i) with two unbounded quantifiers is equivalent to (ii) which starts with a single unbounded quantifier, but which is also Σ_1 – since the bit after the unbounded quantifier, i.e. $(\exists \mathsf{x} \leq \mathsf{w})(\exists \mathsf{y} \leq \mathsf{w}) \varphi(\mathsf{x}, \mathsf{y})$, is of course Δ_0.

The argument then generalizes in entirely obvious ways to cases where φ has additional free variables and/or is preceded by more than two initial unbounded existential quantifiers. Which gives us the desired result. ⊠

10.4 Being more generous with induction

(a) To repeat: (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 which expresses a genuine property/relation. We argued at the beginning of the last section that Δ_0 wffs are eminently suitable, and so far we have only considered instances of induction involving such wffs. It is certainly of some technical interest to see just how many basic truths of arithmetic can in fact be proved by adding that limited amount of induction to Q.[3] But why be so restrictive?

Take *any* open wff φ of L_A at all. This will be built from no more than the constant term '$\mathsf{0}$', the familiar 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 L_A wff at all to be substituted for φ in the Schema?*

Here's a positive argument for generosity. Remember that instances of the Induction Schema (for monadic predicates) are *conditionals* which look like this:

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

So they actually only allow us to derive some $\forall \mathsf{x}\varphi(\mathsf{x})$ when we can *already* prove the corresponding (i) $\varphi(\mathsf{0})$ and also can prove (ii) $\forall \mathsf{x}(\varphi(\mathsf{x}) \to \varphi(\mathsf{Sx}))$. But if we can already prove (i) and (ii) then (iii) we can already prove each and every one of $\varphi(\mathsf{0})$, $\varphi(\mathsf{S0})$, $\varphi(\mathsf{SS0})$, ... However, there are no 'stray' numbers which aren't denoted by some numeral; so that means (iv) that we can prove of each and every number that φ is true of it. What more can it possibly take for φ to express a genuine property that indeed holds for every number, so that (v) $\forall \mathsf{x}\varphi(\mathsf{x})$ is true? In sum, it seems that we can't overshoot by allowing instances of the Induction Schema for *any* open wff φ of L_A with one free variable. The only *usable* instances from our generous range of axioms will be those where we can prove the antecedents (i) and (ii) of the relevant conditionals: and in those cases, we'll have every reason to accept the consequents (v) too. (The argument generalizes in the obvious way to the case where $\varphi(\mathsf{x})$ is relational.)

(b) Suppose we accept that conclusion, and now take it that *any* open wff of L_A can be used in the Induction Schema. This means moving on from Q, and jumping right over a range of possible intermediate theories,[4] to adopt the much richer theory of arithmetic that we can briskly define as follows:

[3] The theory whose non-logical axioms are those of Q plus instances of the Induction Scheme where φ is Δ_0 is commonly known as $\mathsf{I}\Delta_0$ (or $\mathsf{I}\Sigma_0$). One of the first investigations of the scope and limits of $\mathsf{I}\Delta_0$ is by Parikh (1971).

[4] Later, we'll have reason to 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).

> PA – *First-order Peano Arithmetic*[5] – is the first-order theory whose language is L_A and whose axioms are those of Q plus 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 this is a legitimate formalized theory.

Given its very natural motivation, 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. PA allows, in particular, induction for the Σ_1 formula

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

But now note that the corresponding $\varphi(\mathsf{0})$ is a trivial logical theorem. Likewise, $\forall \mathsf{x}\varphi(\mathsf{Sx})$ is an equally 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.[6] 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 ...

10.5 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 intended 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 sentences can be proved in PA: our official choice is just for later metalogical convenience).

And third, its non-logical *axioms* – eliminating the redundancy from our original listing and renumbering – are the following sentences:

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

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

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

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

[5] The name is conventional. Giuseppe Peano did 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).

[6] As we saw in Section 9.9, Axiom 3 enables us to prove some important general claims in Q, despite the absence of the full range of induction axioms. It, so to speak, functions as a very restricted surrogate for induction in certain proofs.

Axiom 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.

10.6 Hoping for completeness?

PA is an intuitively well-motivated theory of basic arithmetic. Investigation and experimentation reveals that, unlike Q, it does indeed have the resources to establish all the familiar elementary general truths about the addition and multiplication of numbers. So we might reasonably have hoped – at least before we'd heard of Gödel's Theorems – that PA would turn out to be a negation complete theory.

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 familiar axioms for successor and addition, plus the universal closures of all instances of the Induction Schema that can be formed in L_P. In short, 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. Note, though, that the availability of a complete formalized theory for successor and addition was proved as early as 1929 by Mojżesz Presburger.[7]

So the situation is as follows, and was known before Gödel got to work. (i) There is a complete formalized theory BA whose theorems are exactly the quantifier-free truths expressible using successor, addition and multiplication (and the connectives). (ii) There is another complete formalized theory (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 in order to get full PA gives us a theory which is incomplete and incompletable (if consistent) comes as a rather nasty surprise. It certainly wasn't obviously predictable that multiplication would make all the

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

To be strictly accurate, Presburger – then a young graduate student – proved the completeness not of P but of a slightly different and less elegant theory (whose primitive expressions were '0', '1' and '+'): see Presburger (1930). But a few years later, Hilbert and Bernays (1934) showed that his methods could be applied to the neater theory P, and it is the latter which is these days typically referred to as 'Presburger Arithmetic'.

difference. Yet it does.[8] As we've already said, *any* consistent axiomatized theory which extends Q will be incomplete.

10.7 Where we've got to

Let's quickly review where we've got to. In Chapter 8 we introduced Robinson Arithmetic Q. We very easily proved this weak theory to be incomplete, using an elementary argument. We just took the sentence $\chi =_{\text{def}} \forall \mathsf{x}(0 + \mathsf{x} = \mathsf{x})$, and showed both that $\mathsf{Q} \nvdash \chi$ and that $\mathsf{Q} \nvdash \neg\chi$ by finding a pair of interpretations which make the axioms of Q all true and respectively make χ and $\neg\chi$ false.

However, we also announced (but did no more than announce) that Q is sufficiently strong. So, assuming it is consistent, Theorem 6.2 will apply and that gives us another, very broadly Gödelian, proof that Q is incomplete. But we aren't yet in a position to *prove* Q's sufficient strength, so for the moment we can't use this more sophisticated route to incompleteness.

In this chapter, we have gone on to introduce the formal arithmetic PA which is the result of adding to Q the universal closure of every instance of the Induction Schema that you can construct in the language L_A. In contrast to Q, this is a very rich and powerful theory, and it certainly isn't *boringly* incomplete – i.e. there is no easy way of coming up with a simple sentence which is demonstrably undecidable in PA. But since PA extends Q and Q is sufficiently strong, PA is sufficiently strong too and Theorem 6.2 applies: hence – assuming that PA is consistent – PA must after all be incomplete. Again, we are not in a position to show that yet, but starting in the next chapter we will begin to put together a version of the original Gödelian argument for this conclusion.

Now, what goes for PA goes for stronger theories. If an axiomatized theory extends PA and remains consistent, it will once more be sufficiently strong, and will be incomplete by Theorem 6.2. Given that our main topic is incompleteness, there is therefore a sense in which we therefore don't really need to give extensive coverage of axiomatized theories stronger than PA; what goes for PA as far as the incompleteness phenomenon is concerned must go for them too.

However, we *will* pause later to say just a little about some of those stronger theories. After all, while PA may be the most natural *first-order* theory of arithmetic, you might well think that it is odd to focus on such a theory. For recall: the intuitive principle of mathematical induction seems to be *second-order*, seems to quantify over properties. Yet despite that, we still proceeded to keep everything within a first-order framework and we introduced the Induction Schema

[8] And by the way, it isn't that multiplication is in itself somehow intrinsically intractable. In 1929 (the proof was published in his 1930), Thoralf Skolem showed that there is a complete theory for the truths expressible in a suitable first-order language with multiplication but lacking addition (or the successor function). Why then does putting multiplication together with addition and successor produce incompleteness? The answer will emerge over the coming chapters, but pivots on the fact that an arithmetic with all three functions built in can express/capture *all* 'primitive recursive' functions.

to handle induction. But why hobble ourselves like that? Why not just add to Q a formal second-order Induction Axiom which directly expresses the intuitive second-order principle, so we get some version of Second-order Peano Arithmetic?

Well, that's a perfectly good question. But it would be too distracting to pause over it now (we'll return to look at second-order arithmetics in Chapter 22). For the moment, first-order PA will serve perfectly well as an example of a very powerful arithmetic which isn't obviously incomplete but which Gödelian arguments show is in fact both incomplete and *incompletable*. No properly axiomatized theory which contains it can be complete: which means that – in particular – properly axiomatized second-order extensions of PA will be incomplete too.

10.8 Is PA consistent?

As we said, PA proves a great deal more than Q. But of course it wouldn't be much joy to discover that PA's power is due to the theory's tipping over into being *inconsistent* and 'proving' *every* wff. So let's end this chapter by briefly considering the issue – *not* because there's a sporting chance that PA might really be in trouble, but because it gives us an opportunity to mention again the consistency theme which we touched on in Section 1.6 and which will occupy us later, from Chapter 24 on.

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 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 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). Therefore (5) the theory is consistent.

This intuitive argument is surely pretty compelling. However, in the end it appeals to our grasp of the structure of the natural numbers and the idea of a numerical property.[9] We are supposed just to *see* that (1) and (2) are true. But

[9] True, we could – if we liked – formalize this intuitive argument inside a rich enough framework such as set theory: but would that actually make it any more compelling? After all, are our intuitions about the basic laws governing the very rich universe of sets *more* secure than those about the basic laws governing the much sparser universe of natural numbers?

now ultra-cautious philosophers will remark that an argument for a theory's consistency which appeals to our supposed intuitive grasp of some intended interpretation *can* lead us badly astray. And to justify their caution, 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 *Grundgesetze der Arithmetik*.[10]

Frege aimed to construct a formal system in which first arithmetic and then the theory of the real numbers can be rigorously developed by deducing them from logic-plus-definitions. He has a wide conception of what counts as logic, which embraces axioms for what is in effect a theory of classes, 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.[11] 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 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. Can't we just 'see' that that's true?

But, famously, the assumption is disastrous (at least when combined with the assumption that classes are themselves things that can belong to classes).[12] As Bertrand Russell pointed out in a letter which Frege received as the second volume of *Grundgesetze* 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 condition R is expressible in the language of Frege's system. So on Frege's assumptions, there will be a class of things that satisfy R. In other words, there is a class Σ_R of all the classes which aren't members of themselves. But now ask: is Σ_R a member of itself? A moment's reflection shows that it is if it isn't, and isn't if it is: contradiction! So there can be no such class as Σ_R; hence Russell's paradox shows that Frege's assumptions cannot be right, despite their intuitive appeal, and his formal system which embodies them is inconsistent.

This sad tale brings home to us vividly that intuitions of truth can be mistaken. But let's not rush to make too much of this. In the end, any argument has to take *something* as given. And the fact that we *can* make mistakes in arguing

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

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

[12]Hence the remark in fn. 9 above, wondering just how secure are our intuitions about the basic laws governing sets.

[13]See Russell (1902).

for the cogency of a formal system on the basis of our supposed grasp of an intended interpretation isn't any evidence at all that we *have* made a mistake in our argument for the consistency of PA. Moreover, 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 appealing to our supposed grasp of an interpretation (or by appealing to a much richer theory like set theory)?'

Yes, there *are* other proofs. However, for now 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 and it will put some interesting limits on the possibilities here.

11 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 13, we'll be able to show that Q and PA in fact do already have the resources to deal with all these functions.

11.1 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 addition and multiplication functions. Informally, and now presented in the style of everyday mathematics (leaving quantifiers to be understood), we have:

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

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

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' – or more precisely, 'primitive recursive'. And our two-clause definitions are examples of *definition by primitive 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 short *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; 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 using a chain of definitions by recursion and composition.[2]

[1] Strictly speaking, we need a proof of the claim that primitive recursive definitions really do well-define functions: such a proof was first given by Richard Dedekind (1888, §126) – for a modern version see, e.g., Moschovakis (2006, pp. 53–56).

There are also other, more complex, kinds of recursive definition – i.e. other ways of defining a function's value for a given argument in terms of its values for smaller arguments. Some of these kinds of definition turn out in fact to be equivalent to definitions by a simple, primitive, recursion: but others, such as the double recursion we meet in defining the Ackermann-Péter function in Section 29.3, are not. For a classic treatment see Péter (1951).

[2] The basic idea is there in Dedekind and highlighted by 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).

That is a quick-and-dirty characterization, though it should be enough to get across the basic idea. Still, 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.

11.2 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 it's because we've already given recursion clauses to define h; or perhaps h is a composite function constructed by plugging one known function into another – as in the case of the factorial, where $h(y, u) = u \times Sy$.

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

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

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 further trivial identity functions that serve to select out particular arguments. Suppose, for example, we have the three-place function $I^3_3(x, y, u) = u$ to hand. Then, in the definition of addition, we can put $h(x, y, u) = SI^3_3(x, y, u)$, so h is defined by composition from previously available functions.

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

(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 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 primitive recursion.*

This covers the case of one-place functions $f(y)$ like the factorial if we allow $\vec{x}$ to be empty, in which case $g(\vec{x})$ is a 'zero-place function', i.e. a constant.

(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 could 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) To summarize. 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 outset, we highlighted the successor function among the given simple functions. But we've

[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.

since noted that, to get our examples to fit our official account of definition by primitive recursion, we need to acknowledge some other, even more trivial, initial functions. So putting everything together, let's now offer this more formal characterization of the p.r. functions (as we'll henceforth call them for short):[4]

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

(We allow g in clauses (2) and (3) to be zero-place, i.e. be a constant.) Note, by the way, that the initial functions are total functions of numbers, defined for every numerical argument; also, 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.

11.3 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 (which makes it transparent that the function *is* 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

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

[5] Compare Section 4.2 where we said that P and Q count as the same property iff 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 – see Section 11.6.

$fermat(n) = n$ 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 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

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

There is no way (algorithmic or otherwise) of settling what Caesar ate on his third birthday! But despite that, the function $julius(n)$ 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 $julius(n)$ must be a p.r. function, though we can't determine *which* function it is from our style of definition.

The salient point is this: primitive recursiveness is a feature of a function itself, irrespective of how it happens to be presented to us.

11.4 The p.r. functions are computable

To repeat, a p.r. function f is one that *can* be specified 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 effectively computable by a simple algorithm. (b) The composition of two computable functions g and h is computable (you just feed the output from whatever algorithmic 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 effectively 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 – as we said – 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 for any number, 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 executing a 'for' loop is that the total number of iterations to be run through 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 using this '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 defined by

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

is calculated by the algorithmic program

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

which gives the value for $f(m, n)$ so long as g and h are computable.

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

factorial ⇒ multiplication ⇒ addition ⇒ successor

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

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

The converse is also true. Take a 'for' loop which computes the value of a function f for given arguments, a loop which calls on two prior routines, one which computes a function g (used to set the value of f with some key argument set to zero), the other which computes a function h (which is used on each loop to fix the next value of f as that argument is incremented). This plainly corresponds to a definition by recursion of f in terms of g and h. And generalizing, 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]

11.5 Not all computable numerical functions are p.r.

We have seen that any p.r. function is mechanically computable. *But not all effectively 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 cook up an example.[7]

First, then, some plausibility considerations. We've just seen that the values of a given primitive recursive function can be computed by a program involving

[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 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] Our cooked-up example, however, isn't one that might be encountered in ordinary mathematical practice: it requires a bit of ingenuity to come up with a 'natural' example – see Section 29.3 where we introduce so-called Ackermann functions.

'for' loops as its main programming structure. Each loop goes through a specified number of iterations. However, back in Section 2.2 we allowed procedures to count as computational even when they don't have nice upper bounds on the number of steps involved. In other words, 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.

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 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.

True, that is as yet only a plausibility consideration. Our remarks so far leave open the possibility that computations can always somehow be turned into procedures using 'for' loops with a bounded limit on the number of steps. But in fact we can now show that isn't the case:

> **Theorem 11.1** *There are effectively computable numerical functions which aren't primitive recursive.*

Proof sketch The set of p.r. functions is effectively enumerable. That is to say, there is an effective way of numbering off 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	$f_3(0)$	$f_3(1)$	$f_3(2)$	$\underline{f_3(3)}$	...
...	...	...	...	...	↘

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 consider our table. Down the table we list off the p.r. functions f_0, f_1, f_2, ... An individual row then gives the values of f_n for each argument. Let's define the corresponding *diagonal* function, by putting $\delta(n) = f_n(n)+1$ (cf. Section 2.3). To compute $\delta(n)$, we just run our effective enumeration of the recipes for p.r. functions until we get to the recipe for f_n. We follow the instructions in that recipe to evaluate that function for the argument n. We then add one. Each step is entirely mechanical. So our diagonal function is effectively computable, using a step-by-step algorithmic procedure.

By construction, however, the function δ can't be primitive recursive. For suppose otherwise. Then δ 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 $\delta(d)$ is. By hypothesis, the function δ is none other than the function f_d, so $\delta(d) = f_d(d)$. But by the initial definition of the diagonal function, $\delta(d) = f_d(d) + 1$. Contradiction.

So we have 'diagonalized out' of the class of p.r. functions to get a new function δ which is effectively computable but 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. definition. And our diagonal argument in effect shows that this can't be done.

11.6 Defining p.r. properties and relations

We have defined the class of p.r. *functions*. Next, we extend the scope of the idea of primitive recursiveness and introduce the ideas of *p.r. (numerical) properties* and *relations*.

Now, quite generally, we can tie talk of functions and talk of properties and relations together by using the 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 (less usual) selection of 0 rather than 1 for *true* is merely 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 (which partition pairs of numbers, etc.). And 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 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]

And, without further ado, we now extend the idea of primitive recursiveness to cover properties and relations:

2. A *p.r. property* is a property with a p.r. characteristic function, and likewise a *p.r. relation* is a relation with a p.r. characteristic function.

Given that any p.r. function is effectively computable, p.r. properties and relations are among the effectively decidable ones.

11.7 Building more p.r. functions and relations

(a) 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 examples of some of these principles at work.

These are more 'trivial but tiresome' details which you could in fact 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 more evidently plausible if you have already worked through some simpler cases. It is therefore probably worth skimming through these sections: but if you have no taste for this sort of detail, don't worry. *Don't get bogged down!*

[8] Compare Section 2.2. 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.

(b) A couple of definitions before the real business gets under way. First, we introduce the *minimization* operator 'μx', to be read: 'the least x such that ...'. Much later, in Section 29.1, we'll be considering the general use of this operator: but here we will be concerned with *bounded minimization*. So we write

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

when f takes the number n as argument and returns as value the least number $x \leq n$ such that $P(x)$ if such an x exists, or returns n otherwise.

Second, suppose that the function f is defined in terms of $k + 1$ 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 mutually 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.

(c) 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.

In each case, the claim amounts to a pretty obvious one about what can be done using 'for' loops. For example, claim (A) comes to this (applied to one-place functions): if you can evaluate $f(m)$ by an algorithmic routine using 'for' loops, then you can check whether $f(m) = n$ by an algorithmic routine using 'for' loops. And claim (C) comes to this (applied to one-place properties): if you can check whether $P(m)$ using 'for' loops, then you can check whether this holds for some or for all $m \leq n$ using 'for' loops. The other claims should all look similarly evident, if you think in terms of what can be computed using 'for' loops, without open-ended searches.

(d) Still, we'd better give official proofs for (A) to (E) for enthusiasts. Note, by the way, that we are doing *informal* everyday mathematics here – i.e. we aren't producing proofs in a formal system, though for brevity's sake we borrow some formal symbols like the connectives and bounded quantifiers.

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

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

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

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

The right-hand side is a composition of p.r. functions, so c is p.r. ⊠

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. Hence 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 any satisfy Px. 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 Px 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, when $k(n)$ 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 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. ⊠

11.8 Further 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.

R1. The relations $m = n$, $m < n$ and $m \leq n$ are primitive recursive.

R2. The relation $m|n$ that holds when m is a factor of n is primitive recursive.

R3. Let $Prime(n)$ be true just when n is a prime number. Then *Prime* is a p.r. property.[9]

R4. List the primes as $\pi_0, \pi_1, \pi_2, \ldots$. Then the function $\pi(n)$ whose value is π_n is p.r.

R5. 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.

R6. 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 unbounded searches, just by using 'for' loops.

(b) We'll now show that those examples are, as claimed, all primitive recursive. However let's stress again that – like the arguments in Section 9.9 – the mini-proofs which follow don't involve anything deep or illuminating. Do skip if you wish!

Proof for (R1) 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 11.1. 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 (R2) 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 (R3) 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 (R4) The function π_n, whose value is the n-th prime (counting from zero), is p.r. – for consider the definition

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

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

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 (R5) 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 (R6) $(Prime(m) \wedge m|n)$ holds when m is a prime factor of n. This is a p.r. relation (being a conjunction of p.r. properties/relations). So it has a p.r. characteristic function which 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.

12 Capturing p.r. functions

At the end of this short 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 are p.r. adequate.

However, we haven't yet explained the idea of capturing a *function* as opposed to capturing a property or relation. So we must start with that.

12.1 Capturing a function

Suppose f is a one-place numerical function. Now define the relation R_f by saying that m has the relation R_f to n just in case $f(m) = n$. 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 $\langle m, f(m)\rangle$.

And just as the characteristic function trick (Section 11.6) 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, as always in this book.)

For a start, consider how we can use this tie to define the idea of *expressing* a function using an open wff. Here is our earlier definition of the idea of expressing a relation, 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:

> 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 $T \vdash \neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$.

Now let's 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 $\varphi(\mathsf{x}, \mathsf{y})$ in theory T 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.

12.2 Two more ways of capturing a function

(a) So far, so good. And that first simple definition could serve us perfectly well. However, two variant notions of capturing functions are also found in the literature (and indeed, are convenient in various ways). We'd better explain these notions too.

Our definition (A) is rather weak. For 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 number m to exactly one value n.

Let's 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 '$\exists!\mathsf{y}$' is the standard uniqueness quantifier, to be read 'there is exactly one y such that ...'.[1] So the new clause (i), as we want, insists that the relation expressed by φ can be proved to correlate any specified number to some unique value. Once again, the generalization of (B) to many-place functions is immediate.

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

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

Proof Assume (i) and (ii) hold. And suppose $f(m) \neq n$ because $f(m) = k$, where $n \neq k$. By (ii), $T \vdash \varphi(\overline{\mathsf{m}}, \overline{\mathsf{k}})$. Hence, by (i), $T \vdash \overline{\mathsf{n}} \neq \overline{\mathsf{k}} \rightarrow \neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$. But as we saw in Section 8.1, even when T is mere Baby Arithmetic, if $n \neq k$, then $T \vdash \overline{\mathsf{n}} \neq \overline{\mathsf{k}}$, and hence $T \vdash \neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$. Which shows that, if T contains BA, (i) and (ii) imply (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, for example, *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 variant definition in the literature, assume that T contains Baby Arithmetic and is consistent. Then the definition in terms of conditions (i) and (ii) is easily seen to be equivalent to this:

> The one-place function f is captured as a function by $\varphi(\mathsf{x}, \mathsf{y})$ 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.

(b) Suppose that $\varphi(\mathsf{x}, \mathsf{y})$ captures the function $f(x) = y$ in theory T. It would be convenient, if only for ease of notation, to expand T's language L to L' by adding a corresponding function symbol 'f', and to expand T to T' by adding a new definitional axiom

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

Then we can say that in T' the function symbol 'f' captures f (shorthand for saying that $\mathsf{f}(\mathsf{x}) = \mathsf{y}$ captures f).

However, augmenting T by what is intended to be a mere notational convenience shouldn't make any difference to which wffs of the original, unextended, language L are provable. In a probably familiar jargon, this extension of T to T' needs to be a *conservative* one. And the condition for it to be conservative to add the function symbol 'f' with its definitional axiom is that T can prove that φ is a functional relation, i.e. 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 pause to prove it here.[2] But note that this *is* a slightly stronger requirement than is given in our idea of φ's capturing f as a function – a weak theory like Q might be able to prove $\exists!\mathsf{y}\varphi(\overline{\mathsf{m}}, \mathsf{y})$ for each number m but be unable to prove the generalization $\forall\mathsf{x}\exists!\mathsf{y}\varphi(\mathsf{x}, \mathsf{y})$.

So we want one more definition. Let's say that

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

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

The generalization to many-place functions is obvious. In summary, if f is fully captured as a function in T, then T either already has a function symbol that captures f, or we can conservatively extend T so that it has one.[3]

12.3 Relating our definitions

That's enough definitions to be going on with! Trivially, if (C) φ fully captures 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, however, do *not* strictly obtain. Still, we have results which are *almost* as good. In particular – and this is the important point for our purposes – if (A) φ captures f in T in the weak sense, then (B) there is a closely related wff $\widetilde{\varphi}$ which *does* capture f *as a function* in T, so long as T is at least as strong as Q.

For suppose that φ captures the one-place function f in the theory T which extends Q. 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. And we can show that this wff not only also captures f but captures it as a function. Why? Essentially because, as we know from Section 9.4, Q (and so any theory T which extends Q) is good at proving results involving bounded quantifiers.

Perhaps we really ought to confirm that claim, so here is the straightforward proof (less enthusiastic readers can skip!):

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

i. for every m, $T \vdash \exists!\mathsf{y}\,\widetilde{\varphi}(\overline{\mathsf{m}},\mathsf{y})$,
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

[3] We have laboured over the A/B/C definitions partly in order to expedite comparisons with discussions elsewhere. But terminology in the literature 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 capturing a function.

So again the moral is plain: when reading other discussions, carefully check the local definitions of the jargon!

$k < n$, $T \vdash \neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{k}})$. 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 (O4) of Section 9.4 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 (O8) of Section 9.4, $\mathsf{a} \leq \overline{\mathsf{n}} \vee \overline{\mathsf{n}} \leq \mathsf{a}$. 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 the string of variables '$\vec{\mathsf{x}}$'), 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 can add that if the theory has a smidgin of induction, then if a function is capturable at all it is *fully* capturable-as-a-function. But we won't prove that further result, but leave it as a testing exercise.

12.4 The idea of p.r. adequacy

(a) The value of a p.r. function for any given argument(s) is computable 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. That motivates a trio of definitions:

> A theory T is *weakly p.r. adequate* (*p.r. adequate*; *strongly p.r. adequate*) if, for every p.r. function f, there is a corresponding φ in T that captures it (respectively: captures it as a function; fully captures it as a function).[4]

(b) Trivially, if T is p.r. adequate in any of those senses, then it also captures every p.r. property and relation. For example, suppose that P is a p.r. property,

[4] Compare the informal idea of being 'sufficiently strong' that we met in Section 6.1. 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.

i.e. a property with a p.r. characteristic function c_P. Since T is p.r. adequate, it captures c_P, which means in particular that there is a wff $\varphi(\mathsf{x}, \mathsf{y})$ such that

if $c_P(m) = 0$, then $T \vdash \varphi(\overline{\mathsf{m}}, \mathsf{0})$
if $c_P(m) \neq 0$, then $T \vdash \neg\varphi(\overline{\mathsf{m}}, \mathsf{0})$

But, by definition, $c_P(m) = 0$ iff Pm, so this amounts to saying that P is captured by $\varphi(\mathsf{x}, \mathsf{0})$.

(c) Now, there's an easy, brute-force, way of constructing a weakly p.r. adequate theory.

Start from BA, our theory of Baby Arithmetic (see Section 8.1). 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 those 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, for example, $\zeta + \xi = \eta$ captures addition in BA.[5]

Next suppose we start expanding BA by adding new vocabulary and new schemata. As a first step, we can add the function symbol '$\uparrow$', intended to express the exponential function, and then say that all numeral instances of the following are axioms too:

Schema 7 $\quad \zeta \uparrow \mathsf{0} = \mathsf{1}$

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

Instances of those schemata enable us to prove the correct result for the value of the exponential function for any arguments, and $\zeta \uparrow \xi = \eta$ captures the exponential in our expanded BA.

For tidiness, let's resymbolize our first four functions using the function symbols '$\mathsf{f_0}$', '$\mathsf{f_1}$', '$\mathsf{f_2}$', '$\mathsf{f_3}$'. And now let's keep going: we will add a symbol '$\mathsf{f_n}$' for each n, with the plan that '$\mathsf{f_n}$' should express the n-th p.r. function f_n in a 'good' effective enumeration of the *recipes* for p.r. functions, where an enumeration is 'good' if there is a p.r. definition of f_n which only involves functions mentioned earlier in the enumeration (particular functions may, of course, be covered repeatedly in the enumeration of recipes, since any p.r. function can be defined in many ways). For each '$\mathsf{f_n}$', we then write down schemata which reflect f_n's recipe defining the function in terms of earlier ones. Call the resulting theory $\mathsf{PRA_0}$.

$\mathsf{PRA_0}$ is still a properly axiomatized theory, because it will be effectively decidable whether any given wff is an instance of one of the axiom schemata. Plainly, its language is much richer than BA's, since it has a separate function symbol 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

[5] Here we take up the permission we gave ourselves in Section 4.5, fn. 5 to read the variables in the official definitions of expressing/capturing as serving as place-holders when necessary.

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 – though, lacking quantifiers, it of course can't be p.r. adequate in the stronger senses.[6]

(d) In sum, we can readily construct a (weakly) p.r. adequate arithmetic by the high-cost method of infinitely expanding the vocabulary of arithmetic and throwing in axioms for every p.r. function. But do we actually *need* to do this?

We don't. In fact, *we only need the language of basic arithmetic in order to frame a (strongly) p.r. adequate theory.* To put it very roughly, the ground we lose by restricting ourselves to a language with successor, addition, and multiplication as the only built-in functions, we can make up again by having quantification available for definitional work. Indeed, even the induction-free arithmetic Q is p.r. adequate. Proving that is work for the next chapter.

[6] '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 is 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 big pay-off is that our later proof of the First Incompleteness Theorem applies even to theories built in the modest language L_A.

13 Q is p.r. adequate

We are now 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. Moreover, it can be captured 'canonically', i.e. by a wff which perspicuously recapitulates the function's definition as a p.r. function.

Here's a road-map of the overall line of argument.

1. *Every* Σ_1 *function can be captured as a function in* Q. A function is Σ_1 if there is a (strictly) Σ_1 wff φ which *expresses* it. We show that we can always massage such a φ into a related Σ_1 wff φ' which *captures* the same function f. We prove this in Section 13.2.

2. *Every p.r. function is a* Σ_1 *function.* This takes us from Sections 13.3 to 13.6 to establish. There are two main phases:

   i. We first use Gödel's 'β-function' trick to prove that L_A has the resources to express any p.r. function f; in effect, we recapitulate within L_A the definition of f by recursion and composition.

   ii. Then 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 indeed Σ_1.

Those two Big Results together immediately entail that Q can capture any p.r. function as a function, i.e. Q is p.r. adequate. It then trivially follows that PA is p.r. adequate too.

This chapter contains the first really heavy-weight proofs in this book. The bad news is that the proofs are significantly tougher going than what's gone before: this is unavoidable. The good news is that the new proof ideas needed to establish our two Big Results are not used again in this book. *So you don't have to master the proofs in this chapter in order to grasp what follows later.* Feel free to skim or even skip. Once more, it is important to say: don't get bogged down in details.

13.1 More definitions

We start with a trio of simple definitions:

> f is a Δ_0 function iff it can be expressed by a Δ_0 wff;
> f is a Σ_1 function iff it can be expressed by a Σ_1 wff;
> f is a Π_1 function iff 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 be using repeatedly. Note too that a function f is Σ_1 as long as it *can* be expressed by some Σ_1 wff: that doesn't rule out its also being expressible in some other way too. For example, we have

Theorem 13.1 *If a function is* Σ_1 *it is also* Π_1.

Proof Suppose the one-place function f can be expressed by the strictly Σ_1 wff $\varphi(\mathsf{x}, \mathsf{y})$. Since f *is* a function, and maps numbers of unique values, we have $f(m) = n$ if and only if $\forall z(f(m) = z \to z = n)$. Hence $f(m) = n$ if and only if $\forall \mathsf{z}(\varphi(\overline{\mathsf{m}}, \mathsf{z}) \to \mathsf{z} = \overline{\mathsf{n}})$ is true. In other words, f is equally well expressed by $\forall \mathsf{z}(\varphi(\mathsf{x}, \mathsf{z}) \to \mathsf{z} = \mathsf{y})$. But it is a trivial exercise of moving quantifiers around to show that if $\varphi(\mathsf{x}, \mathsf{y})$ is strictly Σ_1, then $\forall \mathsf{z}(\varphi(\mathsf{x}, \mathsf{z}) \to \mathsf{z} = \mathsf{y})$ is Π_1. ⊠

13.2 Q can capture all Σ_1 functions

(a) In Section 9.7, we showed that Q can correctly decide every Δ_0 sentence – i.e. can 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 related wff $\widetilde{\varphi}$ (see Section 12.3 again). These results immediately entail

Theorem 13.2 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 13.3 Q *can capture any* Σ_1 *function as a function.*

Proof Our proof falls into two stages.

1. (The really ingenious stage!) We show that any Σ_1 function is equivalent to a composition of two Δ_0 functions.

2. We then show that Q can capture any composition of Δ_0 functions.

Proof of (1) Take the case where we are dealing with a *one*-place Σ_1 function $f(x) = y$. So, by hypothesis, f can be expressed by a strictly Σ_1 open wff with two free variables. And let's assume that this Σ_1 wff has just *one* initial unbounded

quantifier, so it can be rendered as $\exists \mathsf{z} \mathsf{R}(\mathsf{x}, \mathsf{y}, \mathsf{z})$, where $\mathsf{R}(\mathsf{x}, \mathsf{y}, \mathsf{z})$ abbreviates a Δ_0 wff with three free variables.[1] (We'll generalize in a moment.)

R will, of course, express some three-place relation R. So $f(x) = y$ just when $\exists z Rxyz$.

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 a 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. For a given x, $g(x) = c$ puts a ceiling on the numbers we need to search through before finding a pair y, z such that $Rxyz$ is true. And $h(x, c)$ returns a number y under the ceiling c such that for some z also under that ceiling $Rxyz$ is true, or else returns 0. Which means that

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

And the point about this cunning 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,

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

where these wffs are evidently Δ_0.

To complete the proof, we now need to generalize to the cases where we are dealing with a *many*-place Σ_1 function $f(\vec{x}) = y$, and where the strictly Σ_1 wff which expresses f perhaps has the form $\exists \vec{\mathsf{z}} \mathsf{R}(\vec{\mathsf{x}}, \mathsf{y}, \vec{\mathsf{z}})$ with *many* existential quantifiers before the core Δ_0 wff $\mathsf{R}(\vec{\mathsf{x}}, \mathsf{y}, \vec{\mathsf{z}})$. But the generalization is in fact entirely straightforward.[3] ⊠

Proof of (2) 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.

[1] A notational footnote, for the fussy. '$\mathsf{R}(\mathsf{x}, \mathsf{y}, \mathsf{z})$' is serving as an abbreviation for some no doubt long and messy L_A formula. And we said in Section 4.1 we will allow ourselves to use our sans serif font for abbreviations, as well as for the wffs themselves. Here, though, there's some implicitly generalizing work going on, and purists might sternly prefer to use Greek meta-linguistic symbols. However, our notational practice does make for greater readability, since we can exploit the R/R, G/g H/h links that follow. Our notation won't cause any misunderstandings, so – suppressing our purist inclinations – let's stick with it!

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

[3] For the notation '$\vec{x}$' see Section 11.2, (b). Likewise, '$\vec{\mathsf{z}}$' is short for the array of variables $\mathsf{z}_1, \mathsf{z}_2, \ldots, \mathsf{z}_k$, and '$\exists \vec{\mathsf{z}}$' unpacks as a whole bunch of corresponding existential quantifiers, $\exists \mathsf{z}_1 \exists \mathsf{z}_2 \ldots \exists \mathsf{z}_n$. To generalize our argument above, we essentially just need to sprinkle arrows over variables as appropriate!

We'll take a simple case (again, generalizing the argument is easy, and is left as an exercise). So suppose, for example, that

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

where g and h are Δ_0 functions, expressed by the Δ_0 wffs G and H. By the proof of the previous theorem, these functions are therefore captured as functions by the corresponding wffs Δ_0 wffs $\widetilde{\mathsf{G}}$ and $\widetilde{\mathsf{H}}$. We'll now show that the function f is captured by the Σ_1 wff $\mathsf{F}(\mathsf{x}, \mathsf{y}) =_{\text{def}} \exists \mathsf{u}(\widetilde{\mathsf{G}}(\mathsf{x}, \mathsf{u}) \wedge \widetilde{\mathsf{H}}(\mathsf{x}, \mathsf{u}, \mathsf{y}))$.

Suppose that $f(m) = n$. Then for some o, $g(m) = o$, and $h(m, o) = n$. By the capturing assumption, the following are provable in Q (see Section 12.2 (a)):

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

By elementary logic, these together imply

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

So, given our definition of F, it follows that

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

The Σ_1 wff F therefore captures f as a function.

Finishing the proof We have shown that every Σ_1 function can be represented as a composition of two Δ_0 functions, and that a composition of Δ_0 functions can be captured as a function in Q (by a Σ_1 wff). Therefore Q can capture every Σ_1 function as a function (by a Σ_1 wff). ⊠

So stage (1) of this argument tells us how to define two wffs G and H from a given strictly Σ_1 wff φ which *expresses* (but perhaps doesn't capture) some function f. Stage (2) of the argument tells us to form the corresponding wffs $\widetilde{\mathsf{G}}$ and $\widetilde{\mathsf{H}}$ and then put them together into another Σ_1 wff φ' which *captures* the same function f as a function.

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

That's one major theorem under our belt. Our next main aim is to prove

Theorem 13.4 *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). 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 13.4, 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 $\mathsf{I^3_2(x, y, z, u)} =_{\text{def}} (\mathsf{x} = \mathsf{x} \wedge \mathsf{y} = \mathsf{u} \wedge \mathsf{z} = \mathsf{z})$. 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:

$$0! = 1$$
$$(Sx)! = x! \times Sx$$

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 $k_{Su} = k_u \times Su$, 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?

13.4 The idea of a β-function

Let's pause the proof at this point, 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

$$b = \pi_0^{k_0} \cdot \pi_1^{k_1} \cdot \pi_2^{k_2} \cdot \ldots \cdot \pi_n^{k_n}.$$

Like a Gödel 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 from b by using the (primitive recursive) decoding function $exp(b, i)$ which we met in Section 11.8. This function, recall, 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 numerical function $\beta(c, i)$ such that, for *any* finite sequence of natural numbers $k_0, k_1, k_2, \ldots, k_n$ there is a code number c such that for every $i \leq n$, $\beta(b, i) = k_i$.

In other words, for any finite sequence of numbers you choose, you can select a corresponding code number c 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.[4]

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.[5] So the obvious next question is: can we construct a β-function just out of the successor, addition and multiplication functions which *are* built into L_A?

It turns out to simplify things if we liberalize our notion of a β-function just a little. So we'll now also consider *three*-place β-functions, which take *two* code numbers c and d, as follows:

> A three-place β-function is a function of the form $\beta(c, d, i)$ such that, for *any* finite sequence of natural numbers $k_0, k_1, k_2, \ldots, k_n$ there is a pair of code numbers c, d such that for every $i \leq n$, $\beta(c, d, i) = k_i$.

A three-place β-function will do just as well as a two-place function to help us express facts about finite sequences.

Even with this liberalization, though, it still isn't obvious how to define a β-function in terms of the functions built into basic arithmetic. But Gödel neatly solved our problem as follows. Put

[4] 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).

[5] 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 all the fuss in the rest of this section, and in particular 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/PA rather 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.

$$\beta(c, d, i) =_{\text{def}} \text{the remainder left when } c \text{ is divided by } d(i+1)+1.$$

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.[6]

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:

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

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

[6]Here is how to check that claim (this is an exercise in elementary number theory, 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). ⊠

13.5 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 $k_{Su} = k_u \times Su$, 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 $\beta(c, d, Su) = \beta(c, d, u) \times Su$, and $\beta(c, d, x) = y$.

But we've seen that there's a 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. $\exists \mathsf{c} \exists \mathsf{d} \{ \mathsf{B}(\mathsf{c}, \mathsf{d}, 0, 1) \wedge$
$(\forall \mathsf{u} \leq \mathsf{x})[\mathsf{u} \neq \mathsf{x} \rightarrow \exists \mathsf{v} \exists \mathsf{w} \{ (\mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{u}, \mathsf{v}) \wedge \mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{Su}, \mathsf{w})) \wedge \mathsf{w} = \mathsf{v} \times \mathsf{Su} \}] \wedge$
$\mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{x}, \mathsf{y}) \}$.

Abbreviate all that by '$\mathsf{F}(\mathsf{x}, \mathsf{y})$', and we've arrived! For this evidently expresses the factorial function.

Concluding the proof for (3) We need to show that we can use the same β-function trick and prove more generally that, if the function f is defined by recursion from functions g and h which are already expressible in L_A, then f is also expressible in L_A. So here, just for the record, is the entirely routine generalization we need.

We are assuming that

$$f(\vec{x}, 0) = g(\vec{x})$$
$$f(\vec{x}, Sy) = h(\vec{x}, y, f(\vec{x}, y)).$$

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

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

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 each of the claims (1), (2) and (3) from the start of Section 13.3. Hence every p.r. function can be expressed in L_A. Theorem 13.4 is in the bag!

13.6 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* Σ_1 *wff*. Hence

Theorem 13.5 *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 (strictly) Σ_1 wff (C**) 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 p.r. definition of the function expressed).

(b) Now we'll check the key claims (these are just tiresome fiddly details: feel free to skip the rest of this section). Following exactly the same proof strategy that we laid out in Section 13.3, it is evidently 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, so are Δ_0 functions. But as we noted in Section 9.6, every Δ_0 wff is trivially a Σ_1 wff. Hence the initial functions are Σ_1 functions 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 numbers m, n, 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).

So (ii) is true of numbers m, n, just when (i) is, but it 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.[7]

Obviously, this 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

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

where B, remember, is Δ_0. So now consider the wff (C**) constructed as follows. First we drag the quantifier $\exists\mathsf{k}$ and any unbounded existential quantifiers in G to

[7] NB: To drag an existential quantifier forwards across an *unbounded* universal quantifier 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!

the front.[8] We then use the quantifier shift trick to drag the quantifiers $\exists u \exists v$ plus any existential quantifiers embedded in H to the front, moving them past the bounded universal $(\forall u \leq y)$, 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. And 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 f defined by recursion from Σ_1 functions g and h is itself expressible by a strictly Σ_1 wff: in short, f is Σ_1 too. ⊠

13.7 The adequacy theorem

Here are the two Big Results which we've established so far in this chapter:

Theorem 13.3 Q *can capture any* Σ_1 *function as a function.*

Theorem 13.5 *Every p.r. function is* Σ_1.

These of course immediately entail the major theorem that we've been aiming for.[9] Q can capture all p.r. functions as functions, i.e.

Theorem 13.6 Q *is p.r. adequate.*

And by the little result we noted in Section 12.4 (b), that implies that Q can also capture every p.r. property and relation.

Since PA can prove everything Q proves, that means that PA is p.r. adequate too. Indeed, because it contains enough induction, PA is *strongly* p.r. adequate (i.e. can fully capture every p.r. function as a function): but we don't need that result, so we won't pause over it.

13.8 Canonically capturing

We've established that, for any p.r. function f, there exists a wff by which Q can capture it. But our proof doesn't just establish the bald existence claim: it actually shows us how to construct a particular Σ_1 wff that does the capturing job (at least once we are presented with a p.r. definition for f). In a word, we have proved our adequacy theorem *constructively*.

Here's the recipe again, in two main stages. First stage. Take a p.r. definition for f. This involves giving a series of functions f_i ending with f, where each f_i

[8] Changing variables, of course, if that's what it takes to avoid clashes. We won't keep on repeating this sort of caveat.

[9] 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.

is either an initial function or is defined from earlier functions in the series by composition or recursion. Now, for each f_i, we can write down a corresponding Σ_1 wff which expresses it. This is trivial if f_i is an initial function. When f_i is defined by composition from two functions f_j and f_k earlier in the series, we write down a strictly Σ_1 wff expressing f_i which is built by existential quantification from the two Σ_1 wffs which express those earlier functions f_j and f_k (as in the proof of (2′) in Section 13.6). When f_i is defined by recursion from two previous functions f_j and f_k in the series, we write down a strictly Σ_1 wff built on the model of (C**) using the two wffs which express f_j and f_k. So in this way, we eventually get to a strictly Σ_1 wff φ which clearly *expresses* f, essentially by recapitulating a p.r. definition for f.

If we do this first-stage building work carefully, we will in fact already have constructed a wff that captures f. But we haven't officially proved that. Instead – and here's our second stage – we can apply the tricks explained in Section 13.2 to the output of our first-stage construction; for those tricks give us a way of converting any Σ_1 wff φ which expresses f into a corresponding Σ_1 wff φ' which *is* guaranteed to *capture* f.

Even after all this palaver, the wff we end up with still reflects all the details of f's p.r. definition (and we can recover that definition from the wff, thereby revealing the function in question to be p.r.). Let's say that such a capturing wff built up by systematically tracking through f's p.r. definition *canonically captures* f.

Three remarks. (i) Our final constructed wff that canonically captures f of course also *expresses* f. That's obvious just from the way it is constructed – the wff still 'says' the right thing. (Or, since Q is sound, we can use the argument at the very end of Section 4.7 to show that capturing entails expressing.) (ii) There are many ways of canonically capturing a given p.r. function f in Q (and hence in PA). Boringly, we can re-label variables, and shift around 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). (iii) There will be innumerable non-canonical ways of capturing any f: just recall the remark in Section 4.6, (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 as redundant baggage.

14 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 are now behind us. They involved (1) introducing the standard theories Q and PA, then (2) defining the p.r. functions and – the hard bit! – proving Q's p.r. adequacy. In order to do the hard bit, we have already used one elegant idea from Gödel's epoch-making 1931 paper, namely the β-function trick. But most of his proof is still ahead of us: at the end of this Interlude, we'll review the stages that remain.

But first, let's relax for a moment after our labours, and take a very short look at some of the scene-setting background. We'll say more about the historical context in a later Interlude (Chapter 28). But for now, we ought at least to say enough to explain the *title* of Gödel's great paper, 'On formally undecidable propositions of *Principia Mathematica* and related systems I'.[1]

14.1 *Principia*'s logicism

As we noted in Section 10.8, Frege aimed in *Grundgesetze der Arithmetik* to reconstruct arithmetic on a secure footing by deducing it from logic plus definitions. But – in its original form – his overall logicist project flounders on Frege's fifth Basic Law, which leads to contradiction. And the fatal flaw that Russell exposed in Frege's system was not the only paradox to bedevil early treatments of the theory of classes.

Various responses to these contradictions were proposed at the beginning of the twentieth century. One was to try to avoid paradox and salvage logicism by, in effect, keeping much of Frege's logic while avoiding the special assumption about classes 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 the level 1 property to Socrates, i.e. to claim that Socrates is wise. Likewise, the property of *having some instances* is a level 2

[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.

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.

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 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 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 adopt the hierarchical view of classes in their monumental *Principia Mathematica* (1910–13). They take over and develop ('ramify') 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 *Grundgesetze*, and following a similar strategy – to derive all of arithmetic from definitional axioms (compare Section 1.1). 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.)

Let's concentrate, however, on the more limited but still ambitious project of deriving just arithmetic from logic plus definitions.

[2]Exercise: why does this block Russell's paradox from arising? An alternative approach – the now dominant Zermelo-Fraenkel set theory – is much 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 how could anyone think that project is even possible? Well, consider the following broadly Fregean chain of ideas (we will have to ride roughshod over subtleties: but it would be a pity to say nothing, as the ideas are so pretty!).

i. We'll say that the Fs and Gs are *equinumerous* just in case we can match up the Fs and the Gs one-to-one (i.e., in the jargon of Section 2.3, if there is a bijection between the Fs and the Gs). Thus: the knives and the forks are equinumerous if we can pair them up, one to one, with none left over.

   Now, this idea of there being a one-one correspondence between the Fs and the Gs can be defined using quantifiers and identity (so you can see why it might be thought to count as a purely logical notion). Here's one definition, in words: there's a one-one 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 (the function f such that $f(x) = y$ iff Rxy will then be a bijection). In symbols, there's a one-one correspondence between the Fs and the Gs when

$$\exists R\{\forall x(Fx \rightarrow \exists!y(Gy \wedge Rxy)) \wedge \forall y(Gy \rightarrow \exists!x(Fx \wedge Rxy))\}$$

   Here, '$\exists!$' is the uniqueness quantifier again; 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 if 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 C such that the Cs are equinumerous with the Fs*. 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 to the class of all classes with as many members as there are Gs, which holds just if the Fs and Gs are equinumerous.

v. Taking this brute-force line on identifying numbers, we can 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 identical). And pressing on, *one* is the successor of zero. So applying our definitions, it is the class of all classes-of-Fs such

that (i) there is an o which is an F, and (ii) the class of things which are F-but-not-identical-to-o is equinumerous with members of zero. In short – though this makes the story sound circular, which it isn't – one is the class of all one-membered sets. Likewise, *two* is the successor of one: our definitions make two the class of all pairs. And so it goes.

vi. 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 now 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. We have a story about which numbers are natural numbers, and why induction holds. So suppose that you buy the (very big!) assumption that the talk about 'classes' in the story so far still counts as *logical* talk, broadly construed. Then, quite wonderfully, we are launched on our way towards (re)constructing arithmetic in logical terms.

But how do we ensure that we don't run out of natural numbers? If the number n is the class of all n-membered classes, what if there *are* no n-membered classes? Frege answered: there *must* be n-membered classes and so the number n must exist, because the class of *preceding* numbers $\{0, 1, 2, \ldots, n-1\}$ has exactly n members. So given that zero exists, the rest of the natural numbers must exist. Which is ingenious, though – you might well think – a bit *too* ingenious. It smacks suspiciously of conjuring the numbers into existence, one after the other.[3]

In Frege's hands, however, this construction all proceeds inside his flawed logical system. So what if we follow Russell and Whitehead and try to develop a paradox-free version of logicism within *Principia*'s logical system? Well now note that, intuitively, there can be n-membered classes at different levels, and so n (thought of as the class of *all* n-membered classes) would have members at all different levels, which offends against *Principia*'s paradox-blocking stratification of classes into different levels. And Frege's explanation of why we don't run out of numbers also offends.[4] So Russell and Whitehead have to complicate their version of the logicist story considerably. And to ensure that the numbers don't give out they make the bald assumption that there are indeed an unlimited number of things available to be collected into classes of every different finite size. But then this 'Axiom of Infinity' hardly looks like a *logical* assumption.

[3]But for a vigorous neo-Fregean defence, see Wright (1983), Hale and Wright (2001).

[4]If zero, the class containing just the empty class, is two levels up, and the class containing just zero is at level three, then one, conceived of as the class of all classes like $\{0\}$ would be at level four, and there then couldn't be a class containing both zero and one.

However, let's ignore philosophical worries about whether *Principia*'s system counts as pure logic. For now enter Gödel, with a devastating *formal* objection to the idea we can continue the story in any way that would reveal *all* arithmetical truths to be derivable from *Principia*'s assumptions (whether those assumptions count as pure logic or not).

14.2 Gödel's impact

What the First Incompleteness Theorem shows is that, despite its great power, Russell and Whitehead's system still can't capture even all truths of basic arithmetic, at least assuming it *is* consistent. As Gödel puts it:

> 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 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). It perhaps wouldn't be a great surprise, then, to learn that Russell and Whitehead's relatively modest collection of axioms doesn't settle every question that can be posed in their immodest 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*, we *still* don't get the very minimum the logicist hoped to get, i.e. a complete theory of basic arithmetic. (And similarly, there are basic arithmetical propositions which are 'formally undecidable' in ZF set theory.)

14.3 Another road-map

As Gödel himself notes, however, 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 we can now largely forget about *Principia* and pretend henceforth that Gödel was really talking about PA all along.

In what follows, there will be 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 much to be said for introducing the incompleteness theorems by something close to Gödel's own arguments.

Here, then, is an abbreviated reminder of the three stages in our Gödelian proof which remain ahead of us:

1. First, 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 expressions 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 15)

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 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, ¬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 16)

3. As Gödel notes, 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. Which means that PA is not only incomplete but incompletable. Indeed, *any* properly axiomatized theory that contains the weaker theory Q is incompletable. (Chapter 17)

So, map in hand, on we go . . . !

15 The arithmetization of syntax

This chapter falls into three parts. We first introduce Gödel's simple but wonderfully powerful idea of associating the expressions of a formal theory with code numbers. In particular, we'll fix on a scheme for assigning code numbers first to expressions of L_A and then to proof-like sequences of expressions. This double coding scheme will correlate various *syntactic* properties with purely *numerical* properties.

For example, take the syntactic property of being a term of L_A. We can define a corresponding numerical property *Term*, where $Term(n)$ holds just when n codes for a term. Likewise, we can define $Atom(n)$, $Wff(n)$, and $Sent(n)$ which hold just when n codes for an atomic wff, a wff, or a closed wff (sentence) respectively. It will be easy to see that these numerical properties are primitive recursive ones.

More excitingly, we can define the numerical relation $Prf(m, n)$ which holds just when m is the code number in our scheme of a PA-derivation of the sentence with number n. It will also be easy to see – at least in an informal way – that this relation too is primitive recursive.

The second part of the chapter introduces the idea of the *diagonalization* of a wff. This is the idea of taking a wff $\varphi(\mathsf{y})$, and substituting (the numeral for) its own code number in place of the free variable. Think of a code number as a way of referring to a wff. Then the operation of 'diagonalization' allows us to form a wff that as it were indirectly refers to itself (refers to itself via the Gödel coding). We will use this trick in the next chapter to form a Gödel sentence that encodes 'I am unprovable in PA'.

These, then, are the two basically very straightforward ideas to carry away from this chapter. However, we really ought to sketch a proper proof of the crucial claim that *Prf* is indeed primitive recursive. That's the business for the – eminently missable – third part of the chapter.

15.1 Gödel numbering

(a) Here's one way of numbering expressions of L_A. First map individual symbols from L_A's basic alphabet to digits, and then associate a concatenation of symbols with (the number expressed by) the corresponding concatenation of digits. This simple approach is perfectly workable;[1] but we'll in fact use a coding scheme more like Gödel's original version. We'll start by thinking about how to

[1] For example, Raymond Smullyan (1992, pp. 20–24) adopts this kind of coding scheme, which *is* preferable for some purposes. However, our more Gödelian coding scheme will serve us well, and it does – I think – make for some slightly prettier proofs when put to work.

encode expressions of L_A; we'll consider how to code sequences of expressions in the next section.

Suppose that our version of L_A has the usual logical symbols (connectives, quantifiers, identity, brackets), and symbols for zero and for the successor, addition and multiplication functions: associate all those with odd numbers (different symbol, different number, of course). L_A also has an inexhaustible supply of variables, which we'll associate with even numbers. So, to pin that down, let's fix on this preliminary series of *basic codes*:

$\neg$	$\wedge$	$\vee$	$\rightarrow$	$\leftrightarrow$	$\forall$	$\exists$	$=$	(	)	0	S	+	$\times$	x	y	z	...
1	3	5	7	9	11	13	15	17	19	21	23	25	27	2	4	6	...

Our Gödelian numbering scheme for expressions is now defined in terms of this table of basic codes as follows:

> Let expression e be the sequence of $k+1$ symbols and/or variables $s_0, s_1, s_2, \ldots, s_k$. Then e's *Gödel number* (g.n.) is calculated by taking the basic code-number c_i for each s_i in turn, using c_i as an exponent for the $i+1$-th prime number π_i, and then multiplying the results, to get
>
> $$\pi_0^{c_0} \cdot \pi_1^{c_1} \cdot \pi_2^{c_2} \cdot \ldots \cdot \pi_k^{c_k}.$$

For example:

i. The single symbol 'S' has the g.n. 2^{23} (the first prime raised to the appropriate power as read off from our correlation table of basic codes).

ii. The standard numeral SS0 has the g.n. $2^{23} \cdot 3^{23} \cdot 5^{21}$ (the product of the first three primes raised to the appropriate powers).

iii. The wff

$$\exists y\,(S0 + y) = SS0$$

has the g.n.

$$2^{13} \cdot 3^4 \cdot 5^{17} \cdot 7^{23} \cdot 11^{21} \cdot 13^{25} \cdot 17^4 \cdot 19^{19} \cdot 23^{15} \cdot 29^{23} \cdot 31^{23} \cdot 37^{21}$$

That last number is, of course, *enormous*. So when we say that it is elementary to decode the resulting g.n. by taking the exponents of prime factors, we don't mean that the computation is quick and easy. We mean that the computational routine required for the task – namely, repeatedly extracting prime factors – involves no more than the mechanical operations of school-room arithmetic.

(b) Three remarks. First, we earlier allowed the introduction of abbreviatory symbols into L_A (for example, '$\leq$' and '3'); take the g.n. of an expression including such symbols to be the g.n. of the unabbreviated version.

Second, we will later be assuming there are similar numbering schemes for the expressions of other theories which use possibly different languages L. We can imagine each of these numbering schemes to be built up in the same way, but starting from a different table of preliminary basic codes to cope with the different basic symbols of L. We won't spell out the details.

Third, we are going to be introducing numerical properties like *Term* and proving them to be primitive recursive. But note, $Term(n)$ is to hold when n is the code number of a term *according to our Gödel numbering scheme*. However, our numbering scheme was fairly arbitrarily chosen. We could, for example, shuffle around the preliminary assignment of basic codes to get a different numbering scheme; or (more radically) we could use a scheme that isn't based on powers of primes. So could it be that a property like *Term* is p.r. when defined in terms of our arbitrarily chosen numbering scheme and not p.r. when defined in terms of some alternative but equally sensible scheme?

Well, what counts as 'sensible' here? The key feature of our Gödelian scheme is this: there is a pair of *algorithms*, one of which takes us from an L_A expression to its code number, the other of which takes us back again from the code number to the original expression – and moreover, in following through these algorithms, the length of the computation is a simple function of the length of the L_A expression to be encoded or the size of the number to be decoded. The algorithms don't involve open-ended computations using unbounded searches: in other words, the computations can be done just using 'for' loops.

So let S be any other comparable coding scheme, which similarly involves a pair of algorithmic methods for moving to and fro between L_A expressions and numerical codes (where the methods don't involve open-ended searches). And suppose S assigns code n_1 to a certain L_A expression. Consider the process of first decoding n_1 to find the original L_A expression and then re-encoding the expression using our Gödelian scheme to get the code number n_2.[2] By hypothesis, this process will combine two simple computations which just use 'for' loops. Hence, there will be a *primitive recursive* function which maps n_1 to n_2. Similarly, there will be another p.r. function which maps n_2 back to n_1.

Let's say that a coding scheme is *acceptable* iff there is a p.r. function tr which 'translates' code numbers according to S into code numbers under our official Gödelian scheme, and another p.r. function tr^{-1} which converts code numbers under our scheme back into code numbers under scheme S. Then we've just argued that being acceptable in this sense is at least a necessary condition for being an intuitively 'sensible' numbering scheme.[3]

It is immediate that a property like *Term* defined using our scheme is p.r. if and only if the corresponding property $Term_S$ defined using scheme S is p.r., for any acceptable scheme S. Why? Well, let the characteristic functions of *Term*

[2]Strictly, we need to build in a way of handling the 'waste' cases where n_1 isn't an S-code for any wff.

[3]For our purposes, we needn't worry about what might be *sufficient* conditions for being intuitively 'sensible'.

and $Term_S$ be *term* and $term_S$ respectively. Then $term_S(n) = term(tr(n))$, hence $term_S$ will be p.r. by composition so long as *term* is p.r.; and similarly $term(n) = term_S(tr^{-1}(n))$, hence *term* is p.r. if $term_S$ is. So, in sum, *Term* is p.r. iff $Term_S$ is p.r.: the property's status as p.r. is *not* dependent on any particular choice of coding scheme (so long as it is acceptable).

15.2 Coding sequences

As we've already flagged up, the relation $Prf(m, n)$ will be crucial to what follows, where this relation holds just when m codes for an array of wffs that is a PA proof, and n codes for the closed wff (sentence) that is thereby proved. But how *do* we code for proof-arrays?

The details will obviously depend on the kind of proof system we adopt for our preferred version of PA. But we said back in Sections 9.1 and 10.5 that our official line will be that PA has a Hilbert-style axiomatic system of logic. And in this rather old-fashioned framework, proof-arrays are simply *linear sequences* of wffs. A nice way of coding these is by what we'll call *super Gödel numbers*. Given a sequence of wffs or other expressions

$$e_0, e_1, e_2, \ldots, e_n$$

we first code each e_i by a regular g.n. g_i, to yield a sequence of numbers

$$g_0, g_1, g_2, \ldots, g_n.$$

We then encode this sequence of regular Gödel numbers using a single *super g.n.* by repeating the trick of multiplying powers of primes to get

$$2^{g_0} \cdot 3^{g_1} \cdot 5^{g_2} \cdot \ldots \cdot \pi_n^{g_n}.$$

Decoding a super g.n. therefore involves two steps of taking prime factors: first find the sequence of exponents of the prime factors of the super g.n.; then treat those exponents as themselves regular g.n., and take their prime factors to arrive back at a sequence of expressions.

We can now define the proof relation *Prf* more carefully:

> $Prf(m, n)$ holds just if m is the *super* g.n. of a sequence of wffs that is a PA proof of the closed wff with *regular* g.n. n.

15.3 *Term*, *Atom*, *Wff*, *Sent* and *Prf* are p.r.

To repeat: $Term(n)$ is to hold when n codes for a term of L_A. Similarly, $Atom(n)$, $Wff(n)$ and $Sent(n)$ are to hold when n codes for an atomic wff, wff, or sentence of L_A respectively. And we've just defined $Prf(m, n)$. We now need to convince ourselves of the following crucial result:

Theorem 15.1 *$Term(n)$, $Atom(n)$, $Wff(n)$, $Sent(n)$ and $Prf(m, n)$ are primitive recursive.*

And we have a much easier time of it than Gödel did. Writing at the very beginning of the period when concepts of computation were being forged, he couldn't expect his audience to take anything on trust about what was or wasn't '*rekursiv*' or – as we would now put it – primitive recursive. He therefore had to do all the hard work of explicitly showing how to define these properties by a long chain of definitions by composition and recursion.

However, assuming only a very modest familiarity with the ideas of computer programs and p.r. functions, we can perhaps short-cut all that effort and be entirely persuaded by the following:

A very sketchy argument To determine whether $Term(n)$, proceed as follows. Decode n: that's a mechanical exercise. Now ask: is the result a term? That is to say, is it '0', a variable, or built up from '0' and/or variables using just the successor, addition and multiplication functions? (see Section 4.3). That's algorithmically decidable. And both steps involve simple computations that don't involve any open-ended search.

Similarly we can mechanically decide whether $Atom(n)$, $Wff(n)$ or $Sent(n)$. Decode n again. Now ask: is the result an atomic wff, a wff, or a sentence of L_A? In each case, that's algorithmically decidable, without any open-ended searches.

To determine whether $Prf(m, n)$, proceed as follows. First doubly decode m: that's a mechanical exercise. Now ask: is the result a sequence of PA wffs? That's algorithmically decidable (since it is decidable whether each separate string of symbols is a wff). If it does decode into a sequence of wffs, ask: is this sequence a properly constructed proof? That's decidable too (check whether each wff in the sequence is either an axiom or is an immediate consequence of previous wffs by one of the rules of inference of PA's Hilbert-style logical system). If the sequence is a proof, ask: does its final wff have the g.n. n? That's again decidable. Finally ask whether $Sent(n)$ is true. Putting all that together, there is a computational procedure for telling whether $Prf(m, n)$ holds. Moreover, at each and every stage, the computation involved is once more a straightforward, bounded procedure that doesn't involve any open-ended search.

In sum, suppose that we set out to construct programs for determining whether $Term(n)$, $Atom(n)$, $Wff(n)$, $Sent(n)$ or $Prf(m, n)$. Then we will be able to do each of these using programming structures no more exotic than bounded 'for' loops (in particular, we don't need to use any of those open-ended 'do while'/'do until' structures that can take us outside the limits of the primitive recursive).

Now, the computations we've described informally involve shuffling strings of symbols; but – run on a real computer – those will in effect become computations done on binary numbers. And if the whole computation can therefore be done ultimately with 'for' loops operating on numbers, the numerical properties and relations which are decided by the whole procedure must be primitive recursive (see Section 11.4). ⊠

Well, that is indeed sketchy: but the argument ought to strike you as entirely convincing. And if you are happy to take it on trust that we can spell out the details and make all this rigorous, that really is just fine. If you aren't, then Sections 15.6 to 15.9 fill in many of the gaps.

15.4 Some cute notation

That's the first part of the chapter done. Before proceeding to talk about 'diagonalization', let's pause to introduce a really pretty bit of notation. Assume we have chosen some system for Gödel-numbering the expressions of some language L. Then

> If φ is an L-expression, then we'll use '$\ulcorner\varphi\urcorner$' *in our logicians' augmented English* to denote φ's Gödel number.

Borrowing a species of quotation mark is appropriate because the number $\ulcorner\varphi\urcorner$ can be thought of as referring to the expression φ via our coding scheme.

We are also going to use this very same notation to stand in for standard numerals inside our formal language, so that (in our second usage)

> *In abbreviated L-expressions*, '$\ulcorner\varphi\urcorner$' is shorthand for L's standard numeral for the g.n. of φ.

In other words, inside formal expressions '$\ulcorner\varphi\urcorner$' stands in for the numeral for the number $\ulcorner\varphi\urcorner$.

A simple example to illustrate:

1. '$\mathsf{SS0}$' is an L_A expression, the standard numeral for 2.
2. On our numbering scheme $\ulcorner\mathsf{SS0}\urcorner$, the g.n. of '$\mathsf{SS0}$', is $2^{21} \cdot 3^{21} \cdot 5^{19}$.
3. So, by our further convention, we can also use the expression '$\ulcorner\mathsf{SS0}\urcorner$' inside (a definitional extension of) L_A, as an abbreviation for the standard numeral for that g.n., i.e. as an abbreviation for '$\mathsf{SSS}\ldots\mathsf{S0}$' with $2^{21} \cdot 3^{21} \cdot 5^{19}$ occurrences of 'S'!

This double usage – outside a formal language to denote a g.n. of a formal expression and inside a formal language to take the place of a standard numeral for that g.n. – should by this stage cause no confusion at all. (We could have continued our previous practice of overlining abbreviations for standard numerals: we would then indicate the numeral for the g.n. number $\ulcorner\mathsf{SS0}\urcorner$ by the messy '$\overline{\ulcorner\mathsf{SS0}\urcorner}$'. Many writers do this. But aesthetics recommends our fairly common and rather prettier convention.)

15.5 The idea of diagonalization

(a) Gödel is going to tell us how to construct a wff G in PA that is true if and only if it is unprovable in PA. We now have an inkling of how he can do that:

wffs can contain numerals which refer to numbers which – via Gödel coding – are correlated with wffs.

Gödel's construction involves taking an open wff that we'll abbreviate U, or by $\mathsf{U}(\mathsf{y})$ when we want to emphasize that it contains just 'y' free. This wff has g.n. $\ulcorner\mathsf{U}\urcorner$. And then – the crucial move – Gödel substitutes *the numeral for* U*'s g.n.* for the free variable in U. So the key step involves forming the wff $\mathsf{U}(\ulcorner\mathsf{U}\urcorner)$.

This substitution operation is called *diagonalization*, which at first sight might seem an odd term for it. But in fact, Gödel's construction involves something quite closely akin to the 'diagonal' constructions we encountered in e.g. Sections 6.2 and 11.5. In the first of those cases, we matched the index of a wff $\varphi_n(\mathsf{x})$ (in an enumeration of wffs with one free variable) with the numeral substituted for its free variable, to form $\varphi_n(\overline{\mathsf{n}})$. In the second case, we matched the index of a function f_n (in an enumeration of p.r. functions) with the number the function takes as argument, to form $f_n(n)$. Here, in our Gödelian diagonal construction, we match U's Gödel number – and we can think of this as indexing the wff in a list of wffs – with the numeral substituted for its free variable, and this yields the Gödel sentence $\mathsf{U}(\ulcorner\mathsf{U}\urcorner)$.

Now note the following additional point. Given the wff U, it can't matter much whether we do the Gödelian construction by forming (i) $\mathsf{U}(\ulcorner\mathsf{U}\urcorner)$ (as Gödel himself did in 1931) or alternatively by forming (ii) $\exists\mathsf{y}(\mathsf{y} = \ulcorner\mathsf{U}\urcorner \wedge \mathsf{U}(\mathsf{y}))$. For (i) and (ii) are trivially equivalent. But it in fact makes a few technical details go slightly easier if we do things the second way – so that motivates our official definition:

> The *diagonalization* of φ is $\exists\mathsf{y}(\mathsf{y} = \ulcorner\varphi\urcorner \wedge \varphi)$.

It should go without saying that there is no special significance to using the variable 'y' for the significant variable here! But we'll keep this choice fixed, simply for convenience.[4]

(b) Diagonalization is, evidently, a very simple mechanical operation on expressions. In fact,

> **Theorem 15.2** *There is a p.r. function $diag(n)$ which, when applied to a number n which is the g.n. of some wff, yields the g.n. of that wff's diagonalization.*

We will prove this result in the next section; but for the moment here is

Another very sketchy argument Consider this procedure. Decode the g.n. $n = \ulcorner\varphi\urcorner$ to get some expression φ (assume we have some convention for dealing with 'waste' cases where we don't get an expression). Then form φ's diagonalization, $\exists\mathsf{y}(\mathsf{y} = \ulcorner\varphi\urcorner \wedge \varphi)$. Then work out the g.n. of this result to compute $diag(n)$. This

[4] Note too that our official definition of the diagonalization of a wff conveniently applies to any wff, whether it contains 'y' free or not (in the latter case, φ's diagonalization is equivalent to φ).

procedure doesn't involve any unbounded searches. So we again will be able to program the procedure using just 'for' loops. Hence *diag* is p.r. ⊠

15.6 The concatenation function

(a) We have given a very informal but hopefully entirely persuasive argument for Theorem 15.1, and in particular for its crucial claim that the relation *Prf* is primitive recursive, where $Prf(m, n)$ holds when m numbers a PA proof of the sentence with g.n. n.

Gödel, as we said, gives a cast-iron proof of this claim (or rather, he proves the analogue of this claim for his particular formal system). He shows how to define a sequence of more and more complex functions and relations by composition and recursion, eventually leading up to a p.r. definition of *Prf*. Inevitably, this is a laborious job: Gödel does it with masterly economy and compression but, even so, it takes him forty-five steps of function-building to show that *Prf* is p.r.

We have in fact already traced some of the first steps in Section 11.8. We showed, in particular, that extracting exponents of prime factors – the key operation used in decoding Gödel numbers – involves the p.r. function *exp*. To follow Gödel further, we need to keep going in the same vein, defining ever more complex functions. What I propose to do in the third and last part of this chapter is to fill in the next few steps moderately carefully, and then indicate much more briefly how the remainder go. This should be enough to give you a genuine feel for Gödel's demonstration and show you how it can be completed, without going into too much horrible detail.[5]

However, although we are only going to give a partial proof that *Prf* is p.r., by all means skip even this cut-down discussion, and jump to the next chapter where the real excitement starts. You'll miss nothing of wider conceptual interest. Gödel's Big Idea is that *Prf* is primitive recursive. Working up a full proof that this Big Idea is right just involves cracking some mildly interesting brain-teasers, so you might well want to get off the bus at this stop. (Look at it like this. We argued on general grounds in Section 15.3 that *Prf* is p.r.: hence there must be a LOOP program for determining whether $Prf(m, n)$ holds for particular numbers m, n. The rest of this chapter in effect begins to describe how to write the program. Which is fun in its way, if you like that kind of thing. But once you are convinced that the programming tricks *can* be done, you can cheerfully forget *how* they are done.)

(b) Still aboard? Then do some quick revision of Section 11.8. Recall two facts in particular. Keeping the old numbering,

R5. The function $exp(n, i)$ is p.r., where this returns the exponent of π_i in the prime factorization of n.

[5] Masochists and 'completists' are quite welcome to struggle through e.g. Mendelson (1997, pp. 193–198).

R6. The function $len(n)$ is p.r., where this returns the number of distinct prime factors of n.

Note: if n is the g.n. of an expression e which is a sequence of symbols/variables $s_0, s_1, s_2, \ldots, s_k$, then $exp(n, i)$ gives the basic code of s_i. And if n is the super g.n. of a sequence of wffs or other expressions $e_0, e_1, e_2, \ldots, e_k$, then $exp(n, i)$ gives the g.n. of e_i. Further, note that if n is a g.n., then it consists in multiples of the first $len(n)$ primes (i.e. the primes from π_0 to $\pi_{len(n)-1}$).

(c) We will now add another key p.r. function to our list:

R7. There is a *concatenation function* such that (i) $m \star n$ is the g.n. of the expression that results from stringing together the expression with g.n. m followed by the expression with g.n. n, and (ii) this 'star' function is primitive recursive.

Proof for (R7) Suppose m is the g.n. of the expression '$\exists\mathsf{y}$', i.e. $m = 2^{11} \cdot 3^4$, and n is the g.n. of '$\mathsf{y} = \mathsf{0}$', i.e. $n = 2^4 \cdot 3^{13} \cdot 5^{19}$. Then we want $m \star n$ to deliver the g.n. of the concatenation of those two expressions, i.e. the g.n. of '$\exists\mathsf{y}\,\mathsf{y} = \mathsf{0}$'; so we want $m \star n = 2^{11} \cdot 3^4 \cdot 5^4 \cdot 7^{13} \cdot 11^{19}$.

Look at the pattern of exponents here and generalize. Suppose therefore that m and n are Gödel numbers, and that $len(m) = j$ and $len(n) = k$. We want the function $m \star n$ to yield the value obtained by taking the first $j + k$ primes, raising the first j to powers that match the exponents (taken in order) of the j primes in the prime factorization of m and then raising the next k primes to powers that match the k exponents in the prime factorization of n. Then $m \star n$ will indeed yield the g.n. of the expression which results from stringing together the expression with g.n. m followed by the expression with g.n. n.

Now recall that bounded minimization keeps us in the sphere of the primitive recursive (Section 8.5(b)). It is then readily seen we can define a p.r. function $m \star n$ which applies to Gödel numbers in just the right way. We put[6]

$$\begin{aligned} m \star n = (\mu x \leq B_{m,n})[&(\forall i < len(m))\{exp(x, i) = exp(m, i)\} \wedge \\ &(\forall i \leq len(n))\{exp(x, i{+}len(m)) = exp(n, i)\}] \end{aligned}$$

where $B_{m,n}$ has to be a suitable primitive recursive function whose values keep the minimization operator finitely bounded. $B_{m,n} = \pi_{m+n}^{m+n}$ is certainly big enough to cover all eventualities. ⊠

(d) Some mini-examples of the concatenation function at work:

i. $\ulcorner\mathsf{S}\urcorner \star \ulcorner\mathsf{0}\urcorner$ yields $\ulcorner\mathsf{S0}\urcorner$, the g.n. of $\mathsf{S0}$.

ii. Suppose a is the g.n. of the wff $\exists\mathsf{x}\,\mathsf{Sx} = \mathsf{0}$; then $\ulcorner\neg\urcorner \star a$ is the g.n. of $\neg\exists\mathsf{x}\,\mathsf{Sx} = \mathsf{0}$.

[6] A reality check: we are here doing *informal* mathematics. If we use the quantifier and conjunction symbols familiar from our formal languages, that is for brevity's sake. And we can e.g. freely use '$<$' as well as '$\leq$'. These aren't formal wffs! Cf. Section 4.1; also see the comment in the proof for (C) in Section 11.7.

iii. Suppose b is the g.n. of '$\mathsf{S0} = \mathsf{0}$'. Then $\ulcorner(\urcorner \star a \star \ulcorner \rightarrow \urcorner \star b \star \ulcorner)\urcorner$ is the g.n. of $(\exists \mathsf{x}\, \mathsf{Sx} = \mathsf{0} \rightarrow \mathsf{S0} = \mathsf{0})$.

Note, by the way, that $((m \star n) \star o) = (m \star (n \star o))$, which is why we can suppress bracketing with the star function.

(e) Here's a couple more results that we can easily prove now we have the star function to hand:

R8. The function $num(n)$ whose value is the g.n. of the standard numeral for n is p.r.

R9. The function $diag(n)$ from the previous section is p.r.

Proof for (R8) The standard numeral for Sn, for $n > 0$, is of the form 'S' followed by the standard numeral for n. So we have

$$num(0) = \ulcorner \mathsf{0} \urcorner = 2^{19}$$
$$num(Sx) = \ulcorner \mathsf{S} \urcorner \star num(x) = 2^{21} \star num(x).$$

Hence *num* is primitive recursive. ⊠

Proof for (R9) The *diag* function maps n, the g.n. of φ, to the g.n. of φ's diagonalization $\exists \mathsf{y}(\mathsf{y} = \ulcorner \varphi \urcorner \wedge \varphi)$. So we can put

$$diag(n) = \ulcorner \exists \mathsf{y}(\mathsf{y} = \urcorner \star num(n) \star \ulcorner \wedge \urcorner \star n \star \ulcorner)\urcorner$$

where *num* is as just defined, so *diag* is a composition of p.r. functions. ⊠

15.7 Proving that *Term* is p.r.

Prf will evidently be defined in part in terms of *Wff* (since a proof is a sequence of wffs); *Wff* will be defined in terms of *Atom* (since a wff is built up out of atomic wffs); and *Atom* will be defined in terms of *Term* (since an atomic wff is built up out of terms). So we'll begin our outline of a rigorous proof of Theorem 15.1 by showing that *Term* is primitive recursive.

A term, recall, is either '$\mathsf{0}$', or a variable, or is built up from those using the function-symbols 'S', '$+$', '$\times$'. Let's say, then, that a 'constructional history' for a term, or a *term-sequence*, is a sequence of expressions $\langle \tau_0, \tau_1, \ldots, \tau_n \rangle$ such that each expression τ_k in the sequence either (i) is '$\mathsf{0}$'; or else (ii) is a variable; or else (iii) has the form $\mathsf{S}\tau_j$, where τ_j is an earlier expression in the sequence; or else (iv) has the form $(\tau_i + \tau_j)$, where τ_i and τ_j are earlier expressions in the sequence; or else (v) has the form $(\tau_i \times \tau_j)$, where τ_i and τ_j are earlier expressions. Since any well-formed term must have the right kind of constructional history, we can adopt as our official definition: *a term is an expression which is the last expression in some term-sequence.*

So let's now define the numerical relation $Termseq(m, n)$, which holds when m is the super g.n. for a term-sequence, and n is the g.n. of the last expression in that term-sequence. We then have the following results:

R10. The property $Var(n)$ which holds when n is the g.n. of a variable is primitive recursive.

R11. The relation $Termseq(m, n)$ is primitive recursive.

R12. The property $Term$ is primitive recursive.

Proof for (R10) We just note that we can put

$$Var(n) =_{\text{def}} (\exists x \leq n)(n = 2^{2x}).$$

Recall, the basic code for a variable, on our scheme, always has the form $2x$, and the g.n. for a single expression is 2 to the power of the basic code. ⊠

Proof for (R11) We can define $Termseq$ by something of the following shape:

$$Termseq(m, n) =_{\text{def}} exp(m, len(m) - 1) = n \wedge (\forall k < len(m))\{\dots\ exp(m, k)\ \dots\}$$

The *first* conjunct on the right ensures that, in the sequence with super g.n. m, the last expression has g.n. n, as we want. We now need to fill out the curly brackets in the *second* conjunct in a way that reflects the fact for each $k < len(m)$, $exp(m, k)$ – the g.n. of τ_k in our putative term-sequence – is the g.n. of an expression satisfying one of the five defining conditions for belonging to a term-sequence.

So, in those curly brackets, we need to say that $exp(m, k)$ is either (i) the g.n. of '$\mathsf{0}$', or (ii) the g.n. of a variable, or (iii) the g.n. of '$\mathsf{S}\tau_j$' where τ_i occurs earlier in the sequence, or (iv) the g.n. of '$(\tau_i + \tau_j)$' where τ_i and τ_j occur earlier, or (v) the g.n. of '$(\tau_i \times \tau_j)$' where τ_i and τ_j occur earlier.

Here, then, is what we want in the curly brackets:

$$\begin{aligned}\{&exp(m, k) = \ulcorner \mathsf{0} \urcorner \vee \\ &Var(exp(m, k)) \vee \\ &(\exists j < k)(exp(m, k) = \ulcorner \mathsf{S} \urcorner \star exp(m, j)) \vee \\ &(\exists i < k)(\exists j < k)(exp(m, k) = \ulcorner (\urcorner \star exp(m, i) \star \ulcorner + \urcorner \star exp(m, j) \star \ulcorner) \urcorner) \vee \\ &(\exists i < k)(\exists j < k)(exp(m, k) = \ulcorner (\urcorner \star exp(m, i) \star \ulcorner \times \urcorner \star exp(m, j) \star \ulcorner) \urcorner)\}\end{aligned}$$

All the clauses are p.r. conditions, so their disjunction is a p.r. condition, so $Termseq$ is p.r., as we wanted to show. ⊠

Proof for (R12) Since a term has to be the final member of some term-sequence, we can give the following definition:

$$Term(n) =_{\text{def}} (\exists x \leq B_n)\, Termseq(x, n)$$

where B_n is a suitably large p.r. bound. Given a term with g.n. n, and hence with $l = len(n)$ symbols, its term-sequence will be at most l long: so the super g.n. of any term-sequence constructing it must be less than e.g. $B_n = (\pi_l^n)^l$. ⊠

15.8 Proving that *Atom* and *Wff* are p.r.

(a) Given that *Term* is p.r., it follows quickly that

R13. The property $Atom(n)$ which holds when n is the g.n. of an atomic wff is primitive recursive.

Proof for (13) To repeat, the only atomic wffs of L_A are expressions of the kind $\tau_1 = \tau_2$. So we can put

$$Atom(n) =_{\text{def}} (\exists x \leq n)(\exists y \leq n)[Term(x) \wedge Term(y) \wedge n = (x \star \ulcorner = \urcorner \star y)].$$

Which confirms that $Atom(n)$ is primitive recursive. ⊠

(b) We can now proceed to repeat essentially the same tricks to define *Wff*. We first introduce the idea of a *formula-sequence* which is exactly analogous to the idea of a term-sequence. So: a formula-sequence is a sequence of expressions $\langle \varphi_0, \varphi_1, \ldots, \varphi_n \rangle$ such that each φ_k in the sequence is either (i) an atomic wff or else (ii) built from one or two previous wffs in the sequence by using a connective or else (iii) built from a previous wff in the sequence by prefixing $\forall\xi$ or $\exists\xi$ for some variable ξ.[7] We then define $Formseq(m, n)$ to be true when m is the super g.n. of a formula-sequence, and n is the g.n. of the last expression in that sequence. This too is primitive recursive – and to show that it is, we can obviously use exactly the same general strategy as in our proof that *Termseq* is p.r., writing down a disjunction of p.r. conditions. We won't spell out the details here.

Since any wff by definition will be the final formula in a formula-sequence, then we can define the property of numbering a wff:

$$Wff(n) =_{\text{def}} (\exists x \leq B_n) Formseq(x, n).$$

We just need to fix on a p.r. bound B_n within which to look for the formula-sequence for the wff numbered n, and that will confirm that *Wff* is also p.r. (exercise: spell out *Formseq* and find a suitable bound B_n).

15.9 Proving *Prf* is p.r.

(a) And now the end is in sight: we can finally outline a proof that *Prf* is primitive recursive as claimed, and complete the outline Gödelian proof of Theorem 15.1.

The details will evidently depend heavily on the type of proof system we are dealing with. But remember, in order to keep things particularly simple, we're supposing that we have adopted a very old-fashioned linear proof system for PA (which therefore doesn't allow temporary suppositions and sub-proofs). In this kind of system there are propositional axioms such as instances of the schema

[7] A technical remark: things go easiest here if we give L_A a liberal syntax which allows wffs with redundant quantifiers which don't bind any variables in their scope.

$$\varphi \to (\psi \to \varphi)$$

and quantificational axioms such as instances of the schema

$$\forall\xi\varphi(\xi) \to \varphi(\tau) \text{ where } \tau \text{ is a term 'free for' the variable } \xi \text{ in } \varphi(\xi).$$

And we can manage with just two rules of inference, modus ponens and the universal quantification rule that allows us to infer $\forall\xi\varphi$ from φ.[8]

We can then define a PA proof as a sequence of wffs $\langle\varphi_0, \varphi_1, \ldots, \varphi_n\rangle$, each one of which is either an axiom (a PA axiom or a logical axiom), or follows from previous wffs in the sequence by one or other of the rules of inference.

Now, the relation $Modusponens(m, n, o)$ – which holds when the wff with g.n. o follows from the wffs with g.n. m and n by modus ponens – is obviously p.r., for it is immediate that $Modusponens(m, n, o)$ just when

$$m = \ulcorner(\urcorner \star n \star \ulcorner\to\urcorner \star o \star \ulcorner)\urcorner \ \wedge \ Wff(n) \ \wedge \ Wff(o).$$

And the relation $Univ(m, n)$ – which holds when the wff with g.n. n is a universal quantification of the wff with g.n. m – is also obviously p.r., for it is immediate that $Univ(m, n)$ just when

$$(\exists x \leq n)(var(x) \wedge Wff(m) \wedge n = \ulcorner\forall\urcorner \star x \star m).$$

It's trickier – quite a lot trickier, in fact – to prove that the property $Axiom(n)$ is p.r., where $Axiom(n)$ holds just when n numbers a wff which is an axiom (whether a logical axiom or one of the non-logical axioms of PA). That's mainly because, in order to deal with the instances of the logical axiom schemata for quantification, we'll have to arithmetize facts about which variables are free in which wffs, and facts about substituting terms for variables. But for once we won't go into details here: by now you shouldn't need convincing this *can* all be done in a p.r. way. (Rather tough exercise: think how the details might go.)

You shouldn't need convincing either that $Sent(n)$ is also p.r., where $Sent(n)$ holds just when n numbers a sentence, i.e. a closed wff of L_A – for as we noted before, we can mechanically check that an expression is a closed wff without any open-ended searches. (Another tough exercise: think how the details might go for a proof that *Sent* is p.r.)

(b) But if we can assume that $Axiom(n)$ and $Sent(n)$ are p.r., it is easy to get to our final target of showing that the relation $Prf(m, n)$ is indeed primitive recursive.

Proof We echo again the pattern of our definition of *Termseq*. So we want something of the following shape:

[8] See e.g. Mendelson (1997, p. 69): we treat an existential quantifier $\exists\xi$ as mere shorthand for $\neg\exists\xi\neg$. We can in fact recast such a system so that modus ponens is left as the *only* rule of inference, as in the system *QS* of Hunter (1971, pp. 167–168). But this recasting doesn't actually buy us any benefits here.

$$Prf(m,n) =_{\text{def}} exp(m, len(m)-1) = n \ \wedge \ Sent(n) \wedge \\ (\forall k < len(m))\{\dots \ exp(m,k) \ \dots\}.$$

The first conjunct on the right ensures that, in the sequence with super g.n. m, the last expression has g.n. n, and the next conjunct ensures that the sequence finishes with a sentence. We now need to fill out the curly brackets in the second conjunct in a way that reflects the fact for each $k < len(m)$, $exp(m,k)$ – the g.n. of φ_k in our putative proof-sequence – is the g.n. of an expression satisfying one of the defining conditions for belonging to a proof-sequence.

So, in those curly brackets, we need to say that $exp(m,k)$ is either (i) the g.n. of an axiom, or (ii) the g.n. of a wff that follows from two earlier wffs in the sequence by MP, or (iii) the g.n. of a wff which is the universal quantification of some earlier wff.

Putting that in symbols, the curly brackets unpack into

$$\{Axiom(exp(m,k)) \vee \\ (\exists i \leq k)(\exists j \leq k) Modusponens(exp(m,i), exp(m,j), exp(m,k)) \vee \\ (\exists j \leq k) Univ(exp(m,j), exp(m,k))\}.$$

Which is p.r. since it is built up in p.r. ways from p.r. components. ⊠

And that is enough by way of our promised outline proof that $Prf(m,n)$ is primitive recursive!

(c) One very important final comment before leaving this, however. $Prf(m,n)$ holds when m is the super g.n. of a PA proof of the sentence with g.n. n: so $Prov(n) =_{\text{def}} \exists v \, Prf(v,n)$ holds when the wff with g.n. n is provable. Note, though, that this time we *can't* read off from n some upper bound on the length of possible proofs for the sentence with g.n. n. So we *can't* just define the provability property by some *bounded* quantification of the kind $(\exists v \leq B)Prf(v,n)$. If we could, then the provability property would be p.r.: but it isn't – as we will show in Section 21.4.

16 PA is incomplete

The pieces we need are finally all in place. So in this chapter we at long last learn how to construct 'Gödel sentences' and use them to prove that PA is *incomplete*. Then in the next chapter, we show how our arguments can be generalized to prove that PA – and any other formal arithmetic satisfying very modest constraints – is not only incomplete but *incompletable*. Our initial discussion in these two chapters uses ideas from Gödel's own treatment in 1931. Then in Chapter 19, we start extending Gödel's work.

The beautiful proofs now come thick and fast: savour them slowly!

16.1 Reminders

We start with some quick reminders – or bits of headline news, if you have impatiently been skipping forward in order to get to the exciting stuff.

Fix on some acceptable scheme for coding up wffs of PA by using Gödel numbers ('g.n.'), and coding up sequences of wffs by super Gödel numbers. Then,

i. The diagonalization of φ is $\exists \mathsf{y}(\mathsf{y} = \ulcorner\varphi\urcorner \wedge \varphi)$, where '$\ulcorner\varphi\urcorner$' here stands in for the numeral for φ's g.n. – the diagonalization of $\varphi(\mathsf{y})$ is thus equivalent to $\varphi(\ulcorner\varphi\urcorner)$. (Section 15.5)

ii. $diag(n)$ is a p.r. function which, when applied to a number n which is the g.n. of some wff φ, yields the g.n. of φ's diagonalization. (Section 15.6)

iii. $Prf(m, n)$ is a p.r. relation which holds just if m is the super g.n. of a sequence of wffs that is a PA proof of a sentence with g.n. n. (Section 15.3)

iv. Any p.r. function or relation can be *expressed* by a Σ_1 wff of PA's language L_A. (See Section 9.5 for the definition of a Σ_1 wff, and Section 13.6 for the main result.)

v. Any p.r. function or relation can be *captured* in PA by a Σ_1 wff. (Section 13.7)

vi. In particular, we can choose a Σ_1 wff which '*canonically*' captures a given p.r. relation by recapitulating its p.r. definition (or more strictly, by recapitulating the definition of the relation's characteristic function). And this wff which captures the given relation will express it too. (Section 13.8)

For what follows, it isn't necessary that you know the *proofs* of the claims we've just summarized: but do ensure that you at least understand what they say.

16.2 '*G* is true if and only if it is unprovable'

We need just one new result:

> The relation $Gdl(m, n)$ – which holds just when m is the super g.n. for a PA proof of the diagonalization of the wff with g.n. n – is also p.r.

Proof $Gdl(m, n)$ holds, by definition, when $Prf(m, diag(n))$. The characteristic function of Gdl is therefore definable by composition from the characteristic function of Prf and the function $diag$, and hence is p.r., given facts (ii) and (iii) from the last section. ⊠

We can now adopt the following notational convention:

> $\mathsf{Gdl}(\mathsf{x}, \mathsf{y})$ stands in for a Σ_1 wff which canonically captures Gdl,

for we know that there must be such a wff by fact (vi). $\mathsf{Gdl}(\mathsf{x}, \mathsf{y})$ will also express Gdl. And we next follow Gödel in constructing the corresponding wff

$$\mathsf{U}(\mathsf{y}) =_{\text{def}} \forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \mathsf{y}).$$

For convenience, we'll further abbreviate this simply as 'U' when we don't need to stress that it contains 'y' free. Finally we diagonalize U itself, to give

$$\mathsf{G} =_{\text{def}} \exists \mathsf{y}(\mathsf{y} = \ulcorner \mathsf{U} \urcorner \wedge \mathsf{U}(\mathsf{y})).$$

This is our first 'Gödel sentence' for PA.

Trivially, G is equivalent to $\mathsf{U}(\ulcorner \mathsf{U} \urcorner)$. Or unpacking that a bit,

> G is equivalent to $\forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \ulcorner \mathsf{U} \urcorner)$.

It is then immediate that

> G is true if and only if it is unprovable in PA.

Proof By inspection. For consider what it takes for G to be true (on the interpretation built into L_A), given that the formal predicate Gdl expresses the numerical relation Gdl. By our equivalence, G is true if and only if there is no number m such that $Gdl(m, \ulcorner \mathsf{U} \urcorner)$. That is to say, given the definition of Gdl, G is true if and only if there is no number m such that m is the code number for a PA proof of the diagonalization of the wff with g.n. $\ulcorner \mathsf{U} \urcorner$. But the wff with g.n. $\ulcorner \mathsf{U} \urcorner$ is of course U; and its diagonalization is G. So, G is true if and only if there is no number m such that m is the code number for a PA proof of G. But if G is provable, some number would be the code number of a proof of it. Hence G is true if and only if it is unprovable in PA. Wonderful! ⊠

16.3 PA is incomplete: the semantic argument

The very simple proof that, if PA is sound, then PA is incomplete, now runs along the lines we sketched right back in Section 1.2.

Proof Suppose PA is a sound theory, i.e. it proves no falsehoods (because its axioms are true and its logic is truth-preserving). If G (which is true if and only if it is *not* provable) could be proved in PA, the theory *would* prove a false theorem, contradicting our supposition. Hence, G is not provable in PA. Hence G is true. So ¬G is false. Hence ¬G cannot be proved in PA either. In Gödel's words, G is a 'formally undecidable' sentence of PA. ⊠

So that establishes

> **Theorem 16.1** *If* PA *has true axioms, then there is an* L_A*-sentence* φ *such that neither* $\mathsf{PA} \vdash \varphi$ *nor* $\mathsf{PA} \vdash \neg\varphi$.

If we are happy with the semantic assumption that PA's axioms *are* true on interpretation and so PA *is* sound, the argument for incompleteness is as simple as that, once we have constructed G. For reasons that will become clearer when we consider Hilbert's programme and related background in a later Interlude, it was very important to Gödel that incompleteness can *also* be proved *without* supposing that PA is sound: as he puts it, 'purely formal and much weaker assumptions' suffice. However, the further argument that shows this is a little trickier.[1] *So don't lose sight of Gödel's simple 'semantic' argument for incompleteness.*

16.4 '*G* is of Goldbach type'

(a) Before showing how weaker assumptions suffice, let's note that, unlike Theorems 5.7 and 6.1, our new Theorem is proved constructively; in other words, our overall argument doesn't just make the bald existence claim that there is a formally undecidable sentence of PA, it actually tells us how to *construct* one: build the open wff Gdl which canonically captures *Gdl*, then form the corresponding sentence G.

Let's remark too that G, while in one way horribly complex and ridiculously long (when spelt out in unabbreviated L_A), is in another way relatively simple. In the jargon of Section 9.5, G is Π_1.

Proof $\mathsf{Gdl}(\mathsf{x},\mathsf{y})$ is Σ_1. So $\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$ is Σ_1. So its negation $\neg\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$ is Π_1. Hence $\forall\mathsf{x}\neg\mathsf{Gdl}(\mathsf{x},\ulcorner\mathsf{U}\urcorner)$ is Π_1 too. Its logical equivalent G is therefore also Π_1. ⊠

Which means that – as far as its logical complexity goes – the Gödel sentence G is on a par with e.g. a natural formal statement of Goldbach's conjecture, which is another Π_1 sentence, as we noted in Section 9.8.

(b) It is worth pursuing this theme a little further. So note that a Π_1 sentence like Goldbach's conjecture is (equivalent to) the universal generalization of a Δ_0 wff. And any Δ_0 wff expresses a p.r. property or relation. For evidently an atomic

[1] Especially when we move on to consider Rosser's enhanced version of Gödel's argument in Section 19.3, which is needed to get the best non-semantic analogue for Theorem 16.1.

Δ_0 wff expresses a p.r. property or relation, and we know that applying propositional connectives and/or bounded quantifiers keeps us within the realms of the primitive recursive – see Section 11.7. Hence Goldbach's conjecture amounts to a universal generalization about a p.r. property or relation.

That elementary observation motivates a definition:

> A sentence is of *Goldbach type* iff it is (equivalent to) a universal generalization about a p.r. property or relation.

Then what we've just noted is that any Π_1 sentence is of Goldbach type.

The converse holds too: any sentence of Goldbach type is equivalent to a Π_1 sentence.

Proof Take the simplest case, and take a sentence that says that every number has the p.r. monadic property P. Since P is p.r., its characteristic function c_P will be p.r., so c_P is expressible by a Σ_1 wff (by Theorem 13.4 again), and hence also expressible by a Π_1 wff $\varphi(\mathsf{x}, \mathsf{y})$ (by the Σ_1/Π_1 lemma of Section 13.2). So then the Π_1 wff $\varphi(\mathsf{x}, \mathsf{0})$ will express P and the Π_1 sentence $\forall \mathsf{x}\varphi(\mathsf{x}, \mathsf{0})$ will also express the original claim that every number is P. ⊠

(c) In the light of those remarks, rather than talking rather abstractly about our Gödel sentence G being a Π_1 sentence, we can make things quite a bit more vivid by saying that it is of *Goldbach type*, meaning that it is (equivalent to a) universal generalization about a p.r. property. And we can strengthen our statement of Theorem 16.1, to give us

> **Theorem 16.2** *If* PA *has true axioms, then there is an L_A-sentence φ of Goldbach type such that neither* $\mathsf{PA} \vdash \varphi$ *nor* $\mathsf{PA} \vdash \neg\varphi$.

16.5 Starting the syntactic argument for incompleteness

So far, we have actually only made use of the weak result that PA's language can *express* the relation *Gdl*. But in fact our chosen Gdl doesn't just express *Gdl* but (canonically) *captures* it. Using this fact about Gdl, we can again show that PA does not prove G, but this time *without* making the semantic assumption that PA is sound. We'll show that

A. *If* PA *is consistent,* $\mathsf{PA} \nvdash \mathsf{G}$.

Proof Suppose G *is* provable in PA. If G has a proof, then there is some super g.n. m that codes its proof. But by definition, G is the diagonalization of the wff U. Hence, by definition, $Gdl(m, \ulcorner \mathsf{U} \urcorner)$.

Now we use the fact that Gdl captures the relation *Gdl*. That implies that, since $Gdl(m, \ulcorner \mathsf{U} \urcorner)$, we have (i) $\mathsf{PA} \vdash \mathsf{Gdl}(\overline{\mathsf{m}}, \ulcorner \mathsf{U} \urcorner)$.

But since G is logically equivalent to $\forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \ulcorner \mathsf{U} \urcorner)$, the assumption that G is provable comes to this: $\mathsf{PA} \vdash \forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \ulcorner \mathsf{U} \urcorner)$. The universal quantification here entails any instance. Hence (ii) $\mathsf{PA} \vdash \neg \mathsf{Gdl}(\overline{\mathsf{m}}, \ulcorner \mathsf{U} \urcorner)$.

So, combining (i) and (ii), the assumption that G is provable entails that PA is inconsistent. Hence, if PA is consistent, there can be no PA proof of G. ⊠

16.6 ω-incompleteness, ω-inconsistency

Result (A) tells us that if PA is consistent then $\mathsf{PA} \nvdash \mathsf{G}$. There is also a companion result (B), which tells us that if PA satisfies a rather stronger syntactic condition then $\mathsf{PA} \nvdash \neg\mathsf{G}$. This section explains the needed stronger condition of 'ω-consistency'. But first, ...

(a) Here's another standard definition we need:[2]

> An arithmetic theory T is *ω-incomplete* iff, for some open wff $\varphi(\mathsf{x})$, T can prove each $\varphi(\overline{\mathsf{m}})$ but T can't go on to prove $\forall\mathsf{x}\varphi(\mathsf{x})$.

So a theory is *ω-incomplete* if there are cases where it can prove, case by case, that each number satisfies some condition φ but it can't go the extra mile and prove, in one fell swoop, that all numbers satisfy φ. It is an immediate corollary of our last proof that

Theorem 16.3 PA *is ω-incomplete if it is consistent.*

Proof Assume PA's consistency. Then we've shown that

1. $\mathsf{PA} \nvdash \mathsf{G}$ – i.e., $\mathsf{PA} \nvdash \forall\mathsf{x}\neg\mathsf{Gdl}(\mathsf{x}, \ulcorner\mathsf{U}\urcorner)$.

So no number is the super g.n. of a proof of G. That is to say, no number numbers a proof of the diagonalization of U. That is to say, for any particular m, it *isn't* the case that $Gdl(m, \ulcorner\mathsf{U}\urcorner)$. Hence, again by the fact that Gdl captures Gdl, we have

2. For each m, $\mathsf{PA} \vdash \neg\mathsf{Gdl}(\overline{\mathsf{m}}, \ulcorner\mathsf{U}\urcorner)$.

Putting $\varphi(\mathsf{x}) =_{\text{def}} \neg\mathsf{Gdl}(\mathsf{x}, \ulcorner\mathsf{U}\urcorner)$ therefore shows that PA is ω-incomplete. ⊠

In Section 8.3, we noted that Q exhibits a radical case of what we are now calling ω-incompleteness: although it can prove case-by-case all true equations involving numerals, Q can't prove many of their universal generalizations. For a simple example, put $\mathsf{Kx} =_{\text{def}} (\mathsf{0} + \mathsf{x} = \mathsf{x})$; then for every n, $\mathsf{Q} \vdash \mathsf{K}\overline{\mathsf{n}}$, but we don't have $\mathsf{Q} \vdash \forall\mathsf{x}\mathsf{Kx}$. In moving from Q to PA by adding the induction axioms, we vastly increase our ability to prove generalizations. But we now see that some ω-incompleteness must remain even in PA.

(b) And now here's the promised definition of the notion we need for part (B) of our syntactic incompleteness argument:

[2] To help explain choice of the terminology: 'ω' is the logicians' label for the natural numbers taken in their natural order – so, more precisely, ω is the first infinite ordinal.

> An arithmetic theory T is *ω-inconsistent* if, for some open wff $\varphi(\mathsf{x})$, T can prove each $\varphi(\overline{\mathsf{m}})$ and T can also prove $\neg\forall\mathsf{x}\varphi(\mathsf{x})$.

Or equivalently:

> An arithmetic theory T is *ω-inconsistent* if, for some open wff $\varphi'(\mathsf{x})$, $T \vdash \exists\mathsf{x}\varphi'(\mathsf{x})$, yet for each number m we have $T \vdash \neg\varphi'(\overline{\mathsf{m}})$.

Compare and contrast. Suppose T can prove $\varphi(\overline{\mathsf{m}})$ for each m. T is ω-incomplete if it can't also prove something we'd like it to prove, namely $\forall\mathsf{x}\varphi(\mathsf{x})$. While T is ω-inconsistent if it can actually prove the *negation* of what we'd like it to prove, i.e. it can prove $\neg\forall\mathsf{x}\varphi(\mathsf{x})$.

Note that ω-inconsistency, like ordinary inconsistency, is a syntactically defined property: it is characterized in terms of what wffs can be proved, not in terms of what they mean. Note too that ω-consistency – defined of course as not being ω-inconsistent! – implies plain consistency. That's because T's being ω-consistent is a matter of its *not* being able to prove a certain combination of wffs, which entails that T can't be inconsistent and prove *all* wffs.

Now, ω-incompleteness in a theory of arithmetic is a regrettable weakness; but ω-inconsistency is a Very Bad Thing (not as bad as outright inconsistency, maybe, but still bad enough). For evidently, a theory that can prove each of $\varphi(\overline{\mathsf{m}})$ and yet also prove $\neg\forall\mathsf{x}\varphi(\mathsf{x})$ is just not going to be an acceptable candidate for regimenting arithmetic.

(c) That last observation can be made vivid if we bring semantic ideas back into play. Suppose the language of an arithmetic theory T is given a *standard* interpretation, by which we here mean just an interpretation whose domain comprises the natural numbers, and on which T's standard numerals denote the intended numbers (with the logical apparatus also being treated as normal, so that inferences in T are truth-preserving). And suppose further that on this interpretation, the axioms of T are all true. Then T's theorems will all be true too. So now imagine that, for some $\varphi(\mathsf{x})$, T does prove each of $\varphi(\mathsf{0})$, $\varphi(\mathsf{1})$, $\varphi(\mathsf{2})$, ... By hypothesis, these theorems will then be true on the given standard interpretation; so this means that every natural number must satisfy $\varphi(\mathsf{x})$; so $\forall\mathsf{x}\varphi(\mathsf{x})$ is true since the domain contains only natural numbers. Hence $\neg\forall\mathsf{x}\varphi(\mathsf{x})$ will have to be false on this standard interpretation. Therefore $\neg\forall\mathsf{x}\varphi(\mathsf{x})$ can't be a theorem, and T must be ω-consistent.

Hence, contraposing, we have

> **Theorem 16.4** *If T is ω-inconsistent then T's axioms can't all be true on a standard arithmetic interpretation.*

Given that we want formal arithmetics to have axioms which *are* all true on a standard interpretation, we must therefore want ω-consistent arithmetics. And given that we think PA *is* sound on its standard interpretation, we are committed to thinking that it *is* ω-consistent.

16.7 Finishing the syntactic argument

(a) We'll now show that PA can't prove the negation of G, without assuming PA's soundness: we'll just make the syntactic assumption of ω-consistency.

B. *If* PA *is ω-consistent,* $\mathsf{PA} \nvdash \neg\mathsf{G}$.

Proof Suppose that PA is ω-consistent but $\neg$G is provable in PA. That's equivalent to assuming (i) $\mathsf{PA} \vdash \exists \mathsf{x}\mathsf{Gdl}(\mathsf{x}, \ulcorner \mathsf{U} \urcorner)$.

But if PA is ω-consistent, it is consistent. So if $\neg$G is provable, G is *not* provable. Hence for any m, m cannot code for a proof of G. But G is (again!) the wff you get by diagonalizing U. Therefore, by the definition of *Gdl*, our assumptions imply that $Gdl(m, \ulcorner \mathsf{U} \urcorner)$ is false, for each m. So, by the requirement that Gdl captures *Gdl*, we have (ii) $\mathsf{PA} \vdash \neg\mathsf{Gdl}(\overline{\mathsf{m}}, \ulcorner \mathsf{U} \urcorner)$ for each m.

But (i) and (ii) together make PA ω-inconsistent after all, contrary to hypothesis. Hence, if PA is ω-consistent, $\neg$G is unprovable. ⊠

Here's a quick corollary of our argument. Consider the theory $\mathsf{PA}^\dagger = \mathsf{PA} + \neg\mathsf{G}$ (i.e. the theory you get by adding $\neg$G as a new axiom to PA). Then if PA is consistent, this expanded theory $\mathsf{PA}^\dagger$ is consistent but ω-inconsistent.

Proof If $\mathsf{PA}^\dagger$, i.e. PA + $\neg$G, entailed a contradiction, we'd have PA $\vdash$ G, which is impossible if PA is consistent. So if PA is consistent, $\mathsf{PA}^\dagger$ must be consistent too. Further, $\mathsf{PA}^\dagger$ trivially entails $\neg$G, i.e. $\exists \mathsf{x}\mathsf{Gdl}(\mathsf{x}, \ulcorner \mathsf{U} \urcorner)$. And containing PA, it still entails $\neg\mathsf{Gdl}(\overline{\mathsf{m}}, \ulcorner \mathsf{U} \urcorner)$ for each m. So $\mathsf{PA}^\dagger$ is ω-inconsistent. ⊠

Which confirms that ω-inconsistency doesn't imply inconsistency, i.e. that ω-consistency is a stronger requirement than mere consistency.

(b) Let's put all the ingredients together. Recall that G is of Goldbach type. That observation, plus the result (A) from Section 16.5, plus the result (B) which we've just proved, gives us the classic syntactic incompleteness theorem for PA:

> **Theorem 16.5** *There is an L_A-sentence φ of Goldbach type such that, if* PA *is consistent then* $\mathsf{PA} \nvdash \varphi$*; and if* PA *is ω-consistent then* $\mathsf{PA} \nvdash \neg\varphi$.

16.8 'Gödel sentences' and what they say

(a) So much for the two Gödelian proofs of the incompleteness of PA. We can either combine the bolder semantic assumption that PA is sound with the weak result that every p.r. function is expressible in L_A, or combine the more modest syntactic assumption that PA is ω-consistent (and so consistent) with the stronger result that every p.r. function is capturable in PA. Either way, we can show that our G is 'formally undecidable' in PA.[3]

[3]Compare Remark A from the first Interlude commenting on the different assumptions of our two informal arguments for incompleteness in Chapters 5 and 6.

Now, we might call our particular G a *canonical* Gödel sentence for three reasons: (i) it is defined in terms of a wff that we said canonically captures *Gdl*; (ii) because it is roughly the sort of sentence that Gödel himself constructed; (iii) it is the kind of sentence people standardly have in mind when they talk of '*the*' Gödel sentence for PA (though since Gdl certainly isn't unique in canonically capturing *Gdl*, neither is G). However, let's now note that neither our semantic nor our syntactic argument actually depended on fact (i).

To explain. The first, semantic, argument didn't actually rely on the assumption that Gdl perspicuously reflects *Gdl*'s p.r. definition. All we needed was the assumption that we were dealing with *some* wff that expresses *Gdl*, however it does the job. Suppose $\mathsf{Gdl}^{\mathsf{e}}$ is any other wff that expresses *Gdl*. Any wff $\mathsf{Gdl}^{\mathsf{e}}(\mathsf{x},\mathsf{y}) =_{\text{def}} \mathsf{Gdl}(\mathsf{x},\mathsf{y}) \wedge \theta$, where θ is a true sentence, will do – see Section 4.5 (c) – though there are also other, wilder, ways of doing the expressive job. And suppose that G^{e} is defined from $\mathsf{Gdl}^{\mathsf{e}}$ in the same way as G is defined from Gdl. Then, by just the same arguments, G^{e} is also true-but-unprovable in PA, and therefore formally undecidable in PA: so we might naturally call G^{e} a Gödel sentence for PA too. (And if the baggage θ carried along in our sample $\mathsf{Gdl}^{\mathsf{e}}(\mathsf{x},\mathsf{y})$ is Σ_1, then $\mathsf{Gdl}^{\mathsf{e}}(\mathsf{x},\mathsf{y})$ is also Σ_1 – so then G^{e} is still Π_1.)

The second, syntactic, argument likewise didn't actually appeal to the assumption that Gdl canonically captures *Gdl* by recapitulating its p.r. definition. Suppose $\mathsf{Gdl}^{\mathsf{c}}$ is any other wff that captures *Gdl* in PA. For a start, any wff $\mathsf{Gdl}^{\mathsf{c}}(\mathsf{x},\mathsf{y}) =_{\text{def}} \mathsf{Gdl}(\mathsf{x},\mathsf{y}) \wedge \theta$, where θ is a PA theorem, will do – see Section 4.6 (b). And suppose that G^{c} is defined from $\mathsf{Gdl}^{\mathsf{c}}$ in the same way as G is defined from Gdl. Then, again by just the same arguments, G^{c} is also unprovable in PA, assuming the theory is consistent, and $\neg\mathsf{G}^{\mathsf{c}}$ is unprovable, assuming ω-consistency. So we can think of G^{c}-type sentences as a sort of Gödel sentence too. (And if the baggage θ is Σ_1 then G^{c} is again Π_1.)[4]

(b) It is often claimed that a Gödel sentence like G is not only true if and only if unprovable, but actually *says* of itself that it is unprovable: Gödel himself describes his original Gödel sentence this way (Gödel, 1931, p. 151). However, (i) this claim is never *strictly* true; though (ii) if we do restrict ourselves to *canonical* Gödel sentences, then these do *indirectly* say that they are unprovable.

(i) First, let's stress that any of our Gödel sentences (when unpacked) is just another sentence of PA's language L_A, the language of basic arithmetic. It is an enormously long wff involving the first-order quantifiers, the connectives, the identity symbol, and 'S', '+' and '×', which all have the standard interpretation built into L_A. The semantics built into L_A therefore tells us that the various

[4] We think – of course – that PA *is* a sound theory; and we know from Section 4.7 that a wff which captures a relation in a sound theory also expresses it. So each capturing wff $\mathsf{Gdl}^{\mathsf{c}}$ of PA is also an expressing wff $\mathsf{Gdl}^{\mathsf{e}}$; and therefore each formally undecidable Gödel sentence of the type G^{c} is in fact also a Gödel sentence of the type G^{e}, so is true if and only if it is unprovable. But note that this claim *does* depend on assuming PA to be sound. When we generalize our arguments to richer theories which might not be sound, we can't always assume that the second kind of Gödel sentence in fact forms a subclass of the first.

terms inside Gödel sentences (in particular, the numerals) have *numbers* as values. So, their values therefore aren't non-numerical items like, e.g., linguistic expressions. Hence, no Gödel sentence straightforwardly refers to itself: so in particular G doesn't directly refer to itself and say that it is itself unprovable.[5]

(ii) That was trite, but worth stressing. It reminds us that if we claim that some Gödel sentence does 'say' of itself that it is unprovable, then 'say' needs to be carefully qualified.

But now take the case where we are dealing with a canonical Gödel sentence like G. There is a reasonable sense in which this *can* be described as *indirectly* saying that it is unprovable. That, indeed, is the interest in focusing on such canonical Gödel sentences as we have been doing.

Note, this is *not* to make play with some radical re-interpretation of G's symbols (for doing *that* would just make any claim about what G says boringly trivial: if we are allowed radical re-interpretations – like spies choosing to borrow ordinary words for use in a secret code – then any string of symbols can be made to say anything). No, it is because the symbols are still being given their *standard* interpretation that we can recognize that Gdl (when unpacked) will express *Gdl*, given the background framework of Gödel numbering which is involved in the definition of the relation *Gdl*. For remember, Gdl canonically captures *Gdl* and so perspicuously reveals which p.r. function it expresses. Therefore, given that coding scheme, we can recognize just from its construction that G will be true when no number m is such that $Gdl(m, \ulcorner \mathsf{U} \urcorner)$, and so no number numbers a proof of G.

In short, given the coding scheme, *we can immediately see that* G *is constructed in such as way as to make it true just when it is unprovable.* That is the limited sense in which we might reasonably claim that, via our Gödel coding, the canonical G 'indirectly says' that it is unprovable.[6]

[5] The wff G embeds the standard numeral for U(y)'s Gödel number: and given our particular numbering scheme this number will be *huge*. So writing out G in *pure* L_A *without abbreviations* would not be a practical possibility. But how significant is this fact? Well, the situation is actually not at all unusual in mathematics. Given our limited cognitive powers, we need to use chains of abbreviations all the time, and we often work with propositions whose official definitional unpacking into the relevant 'basic' terms would be far too long to grasp. This general phenomenon certainly isn't without its interest and its problematic aspects: see e.g. Isles (1992). However, the phenomenon occurs right across mathematics. So we aren't going to worry unduly that it crops up again here in our dealings with Gödel sentences.

[6] To make the contrast between G and the wider generality of Gödel sentences clear, recall again that if Gdl expresses *Gdl*, so does $\mathsf{Gdl}^{\mathsf{e}}(\mathsf{x}, \mathsf{y}) =_{\text{def}} \mathsf{Gdl}(\mathsf{x}, \mathsf{y}) \wedge \theta$ for any true sentence θ. If θ isn't trivial, so we don't immediately know whether it is true or not, then we won't be able just to read off from its construction that $\mathsf{Gdl}^{\mathsf{e}}$ in fact expresses *Gdl*, in the way that we *can* read that off from the construction of Gdl. Hence, even when you know about the Gödel coding, it won't in general be self-evident that the corresponding G^{e} is true if and only if it is unprovable in PA – that won't be something that G^{e} 'says' on the face of it. See also the discussion in Section 21.2.

17 Gödel's First Theorem

Back in Chapter 8, we introduced the weak arithmetic Q, and soon saw that it is boringly incomplete. Then in Chapter 10 we introduced the much stronger first-order theory PA, and remarked that we couldn't in the same easy way show that it fails to decide some elementary arithmetical claims. However, in the last chapter it has turned out that PA also remains incomplete.

Still, that result in itself isn't yet hugely exciting, even if it is a bit surprising. After all, just saying that a particular theory T is incomplete leaves wide open the possibility that we can patch things up by adding an axiom or two more, to get a complete theory T^+. As we said at the very outset, the real force of Gödel's arguments is that they illustrate *general* methods which can be applied to *any* theory satisfying modest conditions in order to show that it is incomplete. This reveals that a theory like PA is not only incomplete but in a good sense *incompletable*.

The present chapter explains these crucial points.

17.1 Generalizing the semantic argument

In Section 16.3, we showed that PA is incomplete on the semantic assumption that its axioms are true (and its logic is truth-preserving). In this section, we are going to extend this first 'semantic' argument for incompleteness to other theories.

We begin with an important definition. We said in Section 3.3 that a theory T counts as an axiomatized formal theory if it is (i) effectively decidable what counts as a T-wff/T-sentence, (ii) it is effectively decidable which wffs are T's (non-logical) axioms, (iii) T uses a proof system such that it is effectively decidable whether an array of T-wffs counts as a well-constructed derivation, and so (iv) it is effectively decidable which arrays of T-wffs count as proofs from T's axioms. We'll now say that a theory T is *p.r. axiomatized* iff (i′) the numerical properties of being the g.n. of a T-wff/T-sentence are primitive recursive, (ii′) the numerical property of being the g.n. of an axiom is p.r., likewise (iii′) the numerical property of being the super g.n. of a properly constructed proof is p.r., and therefore (iv′) the numerical property of being the super g.n. of a properly constructed proof from T's axioms is p.r. too.[1]

[1]Note, by the way, our remark at the end of Section 15.1, where we explained why a numerical property like being the g.n. of a T-wff will be p.r. on any acceptable numbering scheme if it is p.r. on our default Gödel-style scheme. So the question whether a theory is p.r. axiomatized is not embarrassingly relative to our particular numbering scheme.

Now let's turn to think again about how the semantic argument for PA worked. The key ingredient is the claim that

1. There is an open wff Gdl which *expresses* the relation *Gdl*.

But, let's not forget, we also need another fact:

2. We have quantification available.

For we need quantifiers to form the corresponding wff G from Gdl, where G is true if and only if it is unprovable in PA.[2]

Digging deeper, our demonstration that (1) holds for PA is underpinned by the facts that

3. The relation *Gdl* is primitive recursive (see Section 16.2).

4. PA's language L_A can express all p.r. functions – in fact, express them by Σ_1 wffs (see Section 13.5).

In fact, we can choose a Σ_1 wff which expresses a given p.r. function by perspicuously recapitulating the function's p.r. definition: but that – as we stressed at the end of the last chapter – isn't essential to the argument.

Digging deeper again, *Gdl* can be defined in terms of *Prf* and the trivially p.r. *diag* function. And reviewing our proof that *Prf* is p.r. in Section 15.7, we see that this presupposes that

5. PA is p.r. axiomatized.

For the proof depends crucially on the facts that e.g. *Wff*, *Sent* and *Axiom* are p.r., and that the relations that hold between the codes for the input and output of inference rules are p.r. too.

And that's enough proof-mining to enable us to generalize. Suppose that we are dealing with any other theory T such that

G1. T is p.r. axiomatized;

G2. T's language includes L_A,

where we say that T's language includes L_A if (i) every L_A wff is also a wff of T, perhaps allowing for some definitional extensions of T's original language, and (ii) the copies of L_A wffs in T have the same truth-conditions as before, i.e. the copies are true when their originals are true. Then (G1) gives us the analogue of (3), i.e. there will be a p.r. relation Gdl_T such that $Gdl_T(m, n)$ holds just so

[2] Compare the discussion in Section 12.4, where we outlined the construction of the theory PRA_0 which expresses all p.r. functions (and hence all p.r. properties and relations). This theory can therefore express the version of the p.r. relation *Gdl* defined for that theory, so the analogue of (1) holds. However, the absence of quantifiers in the language of PRA_0 blocks our going on to form a Gödel sentence for that weak theory. Which is why *that* theory, as we claimed, can be negation complete for its limited language.

long as m numbers a T-proof of the diagonalization of the wff numbered by n. And (G2) gives us the analogues of (2) and (4).

There will therefore be an open Σ_1 wff which expresses the p.r. relation $Gdl_T(m, n)$, i.e. the relation which holds just so long as m numbers a T-proof of the diagonalization of the wff numbered by n. Let Gdl_T be such a wff. We can then form a corresponding new Gödel sentence G_T in L_A, again of Goldbach type, which is true if and only if it is unprovable in T.[3]

And from this point on we just use the same easy argument that we used to prove Theorem 16.1 in order to show that, if T is an *arithmetically sound* theory (if its L_A theorems are all true), then G_T is true-but-unprovable, and thus

> **Theorem 17.1** *If a theory T, whose language includes L_A, is p.r. axiomatized and is arithmetically sound, then there is an L_A-sentence φ of Goldbach type such that $T \nvdash \varphi$ and $T \nvdash \neg\varphi$.*

As before, the precise details of the wff Gdl_T are not critical, so long as it does the expressing job. However, we can take the default version of Gdl_T to be one where the wff recapitulates the p.r. definition of Gdl_T. And in this case, as we saw, we get an undecidable sentence G_T which can reasonably be described as 'indirectly saying' of itself that it is unprovable in T.

17.2 Incompletability

Suppose T is a p.r. axiomatized, arithmetically sound theory, with a truth-preserving logic, whose language includes L_A; and suppose φ is one of the undecided arithmetical sentences such that neither $T \vdash \varphi$ nor $T \vdash \neg\varphi$. One of φ, $\neg\varphi$ is true. Consider the result of adding the true one to T as a new axiom. The new expanded theory T^+ is still arithmetically sound, and still a p.r. axiomatized theory, whose language includes L_A. So, although T^+ by construction now decides φ the right way, T^+ *must still be incomplete*.

Take PA as an example. Assume it is sound. Then $\mathsf{PA} + \mathsf{G}$ (the theory you get by adding G as a new axiom) is also sound. This new theory trivially entails G. But being sound, *its* canonical Gödel sentence $\mathsf{G}_{\mathsf{PA}+\mathsf{G}}$ is unprovable in the theory. Of course, the further augmented theory $\mathsf{PA} + \mathsf{G} + \mathsf{G}_{\mathsf{PA}+\mathsf{G}}$ proves it: but being sound, and still p.r. axiomatized, this theory too is incomplete. And so it goes.

Add in as many new true axioms to PA as you like, even augment the truth-preserving deductive apparatus, and the result will still be incomplete – unless

[3] Of course, when we move to consider a different theory T, the set of axioms and/or the set of rules of inference will change (and if T involves new symbols, then the scheme for Gödel-numbering will need to be extended). So the details of the corresponding relation Gdl_T will naturally change too. Hence the details of Gdl_T will change too, and likewise G_T. But still, we can construct our new Gödel sentence along exactly the same general lines as before in constructing the Gödel sentence G for PA; in particular, we'll only need to use basic arithmetical vocabulary.

it ceases to be p.r. axiomatized. In short, PA *is not just incomplete, but it is incompletable* (if we still want a sound, p.r. axiomatized theory).

17.3 Generalizing the syntactic argument

So far, so good. Now we turn to generalizing the syntactic argument for incompleteness. Looking at our proof for Theorem 16.5, we can see that the essential facts underpinning *this* incompleteness proof are:

1′. There is an open wff Gdl which *captures* the relation *Gdl*.

2. We have quantification available.

Again, as we noted, any Gdl that does the capturing job will do. But in fact we took a wff that *canonically* captures *Gdl*.

Underpinning (1′), we have the facts that

3. The relation *Gdl* is primitive recursive.

4′. PA can capture all p.r. functions – in fact, capture them by Σ_1 wffs (see Section 13.7).

But, to repeat, (3) depends essentially on the fact that

5. PA is p.r. axiomatized.

And conditions (2) and (4′) obtain because

6. PA contains Q,

which means (trivially) that PA's language involves quantifiers and (not so trivially) PA can capture all p.r. functions using Σ_1 wffs by Theorem 13.6.

And that's enough proof-analysis for us to be able to generalize again. Suppose then that we are dealing with any other theory T such that

G1. T is p.r. axiomatized.

G2′. T extends Q,

where T extends Q if T's language includes that of Q, and T can prove all Q-theorems.[4] NB, as a limiting case, Q counts as 'extending' itself. Then (G1) gives us the analogue of (3) again: there is a p.r. relation Gdl_T such that $Gdl_T(m, n)$ holds just so long as m numbers a T-proof of the diagonalization of the wff numbered by n. And (G2′) gives the analogue of (4′), so there will be a Σ_1 open wff Gdl_T which captures the relation Gdl_T. Hence, using the familiar construction – and by (G2′), quantification will be available – we can again form a corresponding Gödel sentence G_T.

Here's a useful abbreviatory definition:

[4]See Section 9.8: and for more about this idea, see Section 18.2.

> We'll say that a theory T is *nice* iff T is consistent, p.r. axiomatized, and extends Q.

There doesn't actually seem to be a standard bit of jargon for labelling this kind of theory: so 'nice' is our own local term. Perhaps 'minimally nice' or some such would have been better – since niceness in our sense is only a basic necessary condition for being nice in the everyday sense of being an attractively acceptable theory (for example, a nice theory in our sense can have lots of axioms beyond Q's which are *false* on the standard interpretation). But let's stick to the snappy shorthand.

And once we have constructed the Gödel sentence G_T from Gdl_T, the line of argument will continue exactly as before (as in Sections 16.5 and 16.7), and we get

> **Theorem 17.2** *If T is a nice theory then there is an L_A-sentence φ of Goldbach type such that $T \nvdash \varphi$ and (if T is also ω-consistent) $T \nvdash \neg\varphi$.*

This result obviously gives us a corresponding fact about incompletability. Suppose we beef up some nice, ω-consistent, theory T by adding new axioms. Then T will stay incomplete – unless it becomes ω-inconsistent (bad news) or stops being nice (even worse news).[5]

17.4 The First Theorem

Of course, Gödel in 1931 didn't know that Q is p.r. adequate (because Q wasn't isolated as a minimal p.r. adequate arithmetic until twenty years later). So he didn't have the concept of 'niceness' and it would be anachronistic to identify our very neat Theorem 17.2 as *the* First Incompleteness Theorem. But it is a near miss.

Looking again at our analysis of the syntactic argument for incompleteness, we see that we are interested in theories which extend Q *because we are interested in p.r. adequate theories which can capture p.r. relations like Gdl*. So instead of mentioning Q, let's now instead explicitly write in the requirement of p.r. adequacy. Then we have, by just the same arguments,

> **Theorem 17.3** *If T is a p.r. adequate, p.r. axiomatized theory whose language includes L_A, then there is an L_A-sentence φ of*

[5] A reality check. Since PA doesn't decide every L_A sentence, assuming it is consistent, no *weaker* theory which proves less can decide every L_A sentence either! In other words, it *quite trivially* follows from PA's incompleteness that Q – and every other theory whose language is L_A and which is contained in PA – is incomplete. So the interesting new content of our theorem is that theories *stronger* than PA must stay incomplete too, so long as they are normally axiomatized and remain consistent. We've stated this generalizing theorem as applying to *all* nice theories, rather than as applying just to theories which contain PA, simply to highlight what fundamentally matters in generating incompleteness.

> *Goldbach type such that, if T is consistent then $T \nvdash \varphi$, and if T is ω-consistent then $T \nvdash \neg\varphi$.*

It is this very general syntactic version of the incompleteness result which probably has as much historical right as any to be called Gödel's First Theorem.[6]

For in his 1931 paper, Gödel first proves his Theorem VI, which shows that the formal system P – which is his simplified version of the hierarchical type-theory of *Principia Mathematica* – has a formally undecidable sentence of Goldbach type. Then he immediately generalizes:

> In the proof of Theorem VI no properties of the system P were used besides the following:
>
> 1. The class of axioms and the rules of inference (that is, the relation 'immediate consequence') are [primitive] recursively definable (as soon as we replace the primitive signs in some way by the natural numbers).
> 2. Every [primitive] recursive relation is definable [i.e. is 'capturable'] in the system P.
>
> Therefore, in every formal system that satisfies the assumptions 1 and 2 and is ω-consistent, there are undecidable propositions of the form $(x)F(x)$ [i.e. $\forall x F(x)$], where F is a [primitive] recursively defined property of natural numbers, and likewise in every extension of such a system by a recursively definable ω-consistent class of axioms. (Gödel, 1931, p. 181)

Putting that generalized version together with Gödel's Theorem VIII – which tells us that for any claim $\forall x F(x)$ of Goldbach type there is an equivalent claim which uses only notions from basic arithmetic – gives us our Theorem 17.3.

And so we get to the official First Incompleteness Theorem at very long last. Let joy be unconfined!

[6] 'Hold on! If *that's* the First Theorem, we didn't need to do all the hard work showing that Q and PA are p.r. adequate, did we?' Well, yes and no. No, proving *this* original version of the Theorem of course doesn't depend on proving that any particular theory is p.r. adequate. But yes, showing that this Theorem has real bite, showing that it applies to familiar arithmetics, does depend on proving the adequacy theorem.

By the way, Theorems 17.2 and 17.3 are *not* quite equivalent in effect because there are some alternative very weak p.r. adequate arithmetics which are neither contained in nor contain Q. For some details, see e.g. Boolos et al. (2002, §§16.2, 16.4), comparing the theories which are there called **Q** and **R**. In fact, what's crucial for p.r. adequacy is that **Q** and **R** both deliver all the results that we listed in Section 9.9. But the differences here won't matter to us, and we'll continue to concentrate on Q as the neatest weak p.r. adequate arithmetic.

18 Interlude: About the First Theorem

We have achieved our first main goal, namely to prove Gödel's First Incompleteness Theorem. And it will do no harm to pause for breath and quickly survey what we've established and how we established it. Equally importantly, we should make it clear what we have *not* proved. The Theorem attracts serious misunderstandings. We will briefly block a few of these.

18.1 What we've proved

To begin with the headlines about what we *have* proved (we are going to be repeating ourselves, but – let's hope! – in a good way). Suppose we are trying to regiment the truths of basic arithmetic – i.e. the truths expressible in terms of successor, addition, multiplication, and the apparatus of first-order logic. Ideally, we'd like to construct a consistent theory T whose language includes L_A and which proves *all* the truths of L_A (and no falsehoods). So we'd like T to be negation complete, at least for sentences of L_A. But, given some entirely natural assumptions, there can't be such a negation-complete theory.

The first natural assumption is that T should be set up so that it is effectively decidable whether a putative T-proof really *is* a well-constructed derivation from T's axioms. So, in short, we want T to be a properly axiomatized theory. Indeed, we surely want more: we want it to be decidable what counts as a T-proof without needing open-ended search procedures (if would be a very odd kind of theory where, e.g., checking whether some wff is an axiom takes an unbounded search). Sharpened up, this is the assumption that T should be not just an axiomatized formal theory but be a p.r. axiomatized theory.

But next, there's a fork in the path.

(a) Let's first assume that T is an *arithmetically sound* theory – after all, our original motivation for being interested in the issue of completeness was wanting a theory that regiments all the *truths* of L_A. Given the soundness assumption, we then have the following *semantic* argument for incompleteness (remember, we are assuming T's language includes L_A):

1. Every p.r. function and relation can be *expressed* in L_A, the language of basic arithmetic. (Theorem 13.4, proved in Sections 13.3 to 13.5 – the argument is elementary once we grasp the β-function trick.)

2. The relation $Prf_T(m, n)$, which holds when m codes for a T-proof of the wff with code number n, is primitive recursive – on the assumptions

that T is p.r. axiomatized and that we have a sensible coding system. (That's Theorem 15.1, for which we gave a quick and dirty but convincing argument in Section 15.3.) Likewise, the relation $Gdl_T(m, n)$, which holds when m codes for a T-proof of the diagonalization of the wff with code number n, is primitive recursive (since Gdl_T is definable in terms of Prf_T and the trivially p.r. $diag_T$ function).

3. Then, using the fact that Gdl_T must be expressible in L_A, and hence also in the language of T, we can construct an arithmetic wff G_T which is true if and only if it is unprovable. A Π_1 wff, i.e. one of Goldbach type, suffices for the job. (See section 16.2: the construction is ingenious, but quite easy to understand.)

4. Since G_T is true if and only if it is unprovable, then – given the assumption that T is arithmetically sound – there's an entirely straightforward argument to the dual conclusions that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$. (Use the argument which we first gave, in fact, in Section 1.2.)

In sum, then, Gödel's semantic argument shows that

> **Theorem 17.1** *If a theory T, whose language includes L_A, is p.r. axiomatized and is arithmetically sound, then there is an L_A-sentence φ of Goldbach type such that $T \nvdash \varphi$ and $T \nvdash \neg\varphi$.*

And now let's now quickly compare this result with the upshot of the easy semantic incompleteness proof that we gave much earlier:

> **Theorem 5.7** *If T is a sound axiomatized theory whose language is sufficiently expressive, then T cannot be negation complete.*

What has all our hard work since Chapter 5 bought us?

One point of difference between the old and new results is that the old result applied to axiomatized theories generally, and the new result is so far only about p.r. axiomatized theories in particular. But that's minor. We've just noted that we are normally interested in axiomatized theories that are presented as being p.r. axiomatized: and, as we'll see in Section 19.1, *any* decidably axiomatized theory can in fact be recast as a p.r. axiomatized one.

So the key point of difference between the old and new results is this. The old theorem told us about theories with 'sufficiently expressive' languages (i.e. languages that can express all decidable two-place numerical relations), but it didn't tell us anything at all about what such languages look like. The possibility was left open that sufficiently expressive languages would have to be quite rich.[1] And that in turn left open the possibility that – even though a theory with a richly expressive language can't be complete – weaker theories covering e.g. basic

[1] Though we did say something to motivate the thought that sufficiently expressive languages don't have to be *very* rich in Section 5.1, fn. 1.

arithmetic could be complete. For remember: you can have incomplete theories which extend complete sub-theories (e.g., Q extends BA, which is a complete theory of the truths in its cut-down language L_B.) Our new result closes off those possibilities: even theories built in L_A must be incomplete.

(b) So much for Gödel's *semantic* route to his incompleteness result. Now suppose, however, we take the other fork in the path, and don't assume that T is sound. We make life rather harder for ourselves. But we can still prove that T must be incomplete, given weaker assumptions. We can put one version of the *syntactic* argument as follows:

1′. Assume T contains at least as much arithmetic as Q. Then every p.r. function and relation can be *captured* in T. (Theorem 13.6, which rests on two other results: (i) every p.r. function is expressible using a Σ_1 wff of L_A; and (ii) Q can capture any function expressible by a Σ_1 wff.)

2′. As before, assuming that T is p.r. axiomatized, Prf_T and so $Gdl_T(m, n)$ are primitive recursive.

3′. Again, we can construct the wff G_T in the now familiar way, starting from a wff that captures Gdl_T.

4′. We can then show: if T is consistent, $T \nvdash \mathsf{G}_T$; and if T is ω-consistent, $T \nvdash \neg\mathsf{G}_T$. (Generalizing the arguments of Sections 16.5 and 16.7.)

Note that the stages 1′ and 4′ of the syntactic argument are a bit trickier than their counterparts in the semantic argument.[2] That's the price we pay for managing without the strong assumption of soundness. But we get a benefit, for we of course get an incompleteness result that now applies irrespective of the soundness of the theory in question:

> **Theorem 17.2** *If T is a nice theory – is a consistent, p.r. axiomatized theory which extends* Q *– then there is an L_A-sentence φ of*

[2]Trickier, yes. But still not *that* difficult. And it was very important to Gödel's overall conception of his enterprise that this was so. Georg Kreisel puts it like this in his memoir of Gödel:

> Without losing sight of the permanent interest of his work, Gödel repeatedly stressed – at least, during the second half of his life – how little novel mathematics was needed; only attention to some quite commonplace (philosophical) distinctions; in the case of his most famous result: between arithmetic truth on the one hand and derivability by (any given) formal rules on the other. Far from being uncomfortable about so to speak getting something for nothing, he saw his early successes as special cases of a fruitful general, but neglected, scheme:
>
> By attention to or, equivalently, analysis of suitable traditional philosophical notions and issues, adding possibly a touch of precision, one arrives painlessly at appropriate concepts, correct conjectures, and generally easy proofs. (Kreisel, 1980, p. 150)

> *Goldbach type such that $T \nvdash \varphi$ and (assuming T is ω-consistent) $T \nvdash \neg\varphi$.*

Let's again quickly compare this result with the upshot of an earlier incompleteness argument, this time the 'syntactic' proof that:

> **Theorem 6.2** *A consistent, sufficiently strong, axiomatized formal theory of arithmetic cannot be negation complete.*

What has all our hard work since Chapter 6 bought us this time?

Well, one point of difference between the old and new results is again that the old one applied to axiomatized formal theories while the new result so far applies only to p.r. axiomatized theories. But that's not important: and, as we said, we'll soon be closing this gap anyway.

Another, more significant, point of difference is that the old result required only the assumption that our theory T is consistent, while half the new result requires the stronger assumption that T is ω-consistent. But in fact, this too is a difference which we can with some effort massage away. We can replace G_T with a cunningly designed sentence R_T, and replace the relatively easy argument for G_T's undecidability (assuming T's ω-consistency) with a correspondingly more complex argument for R_T's undecidability (assuming only that T is consistent). This will enable us to show that nice theories are *always* incomplete, whether ω-consistent or not. (For more on this, see the Gödel-Rosser Theorem in Section 19.3.)

The critical difference between our old and new syntactic arguments for incompleteness is therefore this. The old theorem told us about theories which are 'sufficiently strong' (i.e. theories which can capture all decidable properties of numbers), but it didn't tell us what such theories look like. So the possibility was left open that such incomplete theories have to be immensely rich if they are to capture all decidable properties. Our new result has foreclosed that possibility too. It tells us that to get incompleteness via the syntactic route it is enough to be working with a theory which captures all p.r. properties and relations – and it suffices that the theory includes Q.

(c) Let's summarize the summary! Beginning with the First Interlude, we have repeatedly stressed that there are *two* routes to incompleteness results. One route goes via the semantic assumption that we are dealing with sound theories, and otherwise uses a weak result about what certain theories can *express*: the other route goes via the syntactic assumptions of consistency/ω-consistency, and has to combine that with a stronger result about what theories can *prove*.

In his 1931 paper, Gödel downplays the semantic route to incompleteness. In fact, it is indicated only at the outset, in his informal introduction, where he 'sketch[es] the main idea of the proof ... without any claim to complete precision'. In Chapter 28, we will say something about why Gödel wants officially to avoid relying on the notion of truth in his paper, and why he highlights the syntactic

route. But it is worth bearing in mind that there really is a matching pair of results here.

18.2 The reach of Gödelian incompleteness

Our generalized theorems talk about theories T whose language 'includes' L_A. We now need to be a bit more explicit about that notion.

Suppose T's 'native' language initially contains no arithmetical vocabulary. In that case, for the theorems to apply, T needs to be able to define equivalents to L_A-wffs. But that involves *two* things. First, T must of course be able to define the basic arithmetical vocabulary '$\mathsf{0}$', 'S', '$+$' and '$\times$'. But also T needs to be able to define a predicate 'Nat' (satisfied by whatever play the role of natural numbers in T's domain of quantification): that's to enable T to replicate L_A's arithmetical quantifications.

Take, for example, an L_A-wff of the form $\forall \mathsf{x}\varphi(\mathsf{x})$, where φ involves no quantifiers. It plainly *won't* do to take the equivalent claim in T just to be something of the form $\forall \mathsf{x}\varphi'(\mathsf{x})$, where φ' is T's equivalent to φ according to T's definition of the arithmetical vocabulary. That's because T's quantifiers might range more widely that L_A's. To get the 'native' equivalent of the L_A quantified claim, we need to restrict the T's quantifiers to the domain of numbers: so T's version of the L_A claim will unpack into something of the form $\forall \mathsf{x}(\mathsf{Nat}(\mathsf{x}) \to \varphi'(\mathsf{x}))$.

And now the point to emphasize is that even if the theory T talks about a domain which contains (an analogue of) the natural numbers, T may lack a way of picking them out, i.e. T may lack a way of defining a suitable predicate Nat. In that case, T *won't* be able to prove or even express e.g. the analogues of Q's quantified theorems, and then version of the incompleteness theorem can get a grip.

Here's an illustration of the point (don't worry if the example is unfamiliar: for our purposes, it's only the general moral that matters). Compare Q, an axiomatic theory of rational fields ('fractions'), and R, the textbook axiomatic theory of real closed fields ('real numbers'). Both theories are only true of domains big enough to contain a '0', a '1', and all their successors; i.e. both theories are only true of domains which contain (something that looks like) the natural number series. Now, Julia Robinson (1949) showed that you can indeed construct a Q-predicate Nat which is only true of the natural numbers – though doing this takes some surprisingly sophisticated ideas. Hence, you can construct a theory of the natural numbers inside Q: so Q extends Q and must be *incomplete* and incompletable too. On the other hand, Alfred Tarski had earlier proved that R is a *complete* theory – which means that there can't be a predicate of R which picks out the natural numbers. Put it this way: while the real numbers contain the natural numbers, the pure theory of real numbers doesn't contain the theory of natural numbers.[3] The moral? Even if a theory T deals with a more complex

[3]For more on Tarski's theorem here, see e.g. Hodges (1997, Sections 2.7, 7.4).

structure than the natural numbers, the incompleteness of arithmetic doesn't by itself entail the incompleteness of T.

18.3 Some ways to argue that G_T is true

The Gödelian arguments show that if a rich enough arithmetical theory T is sound, or indeed if it is just consistent, then there will be a canonical Gödel sentence G_T which is unprovable in T, and – because it indirectly 'says' it is unprovable – G_T will be true. It is now worth pausing to think just a bit about the variety of situations in which we might come to recognize that G_T *is* true.

Start with the most humdrum case. (1) I believe that G_Q is true. Why? Well, I endorse Q as a good (though very partial) arithmetical theory. I'm very happy to assert its theorems. And standing back from my cheerful willingness to follow derivations in Q where they lead, I recognize that I accept the theory as sound. Then the semantic argument for incompleteness tells me that G_Q is unprovable and hence true.

(2) A fancier case. Continue to assume that Q is sound, and now consider the theory Q_G that we introduced in Section 9.8, which is the theory we get by adding Goldbach's conjecture as an additional axiom to Q. Then Q_G is consistent if and only if Goldbach's conjecture is true of the natural numbers (the 'if' direction is trivial, for if the conjecture is true all Q_G axioms are true; the 'only if' direction is – as we noted – a corollary of Theorem 9.4). We should therefore accept the canonical Gödel sentence for Q_G as being true if and only if we in fact accept Goldbach's conjecture itself as true. And why do we accept *that*, if we do? In part because of our failure ever to find any counterexamples to the conjecture. And in part, perhaps, because of some probabilistic considerations concerning the distribution of primes, and other 'almost-proofs' for the conjecture.

Here's a variant example (2′). This time take the theory Q_F which we get by adding a statement of Fermat's Last Theorem as an additional axiom to Q. Since we can regiment Fermat's Last Theorem as a Π_1 wff, then – by Theorem 9.4 again – Q_F is consistent if and only if Fermat's Last Theorem is true. We more confidently believe *this* theory is consistent because we have Andrew Wiles's (highly infinitary) *proof* of the Theorem. And that proof then grounds our belief that Q_F's canonical Gödel sentence must be true.

Consider next (3) the case of the theory $\mathsf{PA}^\dagger$ which you get by adding $\neg\mathsf{G}$ to PA as a further axiom. This theory is unsound on the interpretation built into its language L_A. But it must be consistent, assuming PA is, for otherwise we'd have $\mathsf{PA} \vdash \mathsf{G}$, contrary to the Incompleteness Theorem for PA. So, given we accept that PA *is* consistent and know about Gödel's syntactic incompleteness proof, we'll accept that $\mathsf{PA}^\dagger$ is consistent, and hence also accept the canonical Gödel sentence for $\mathsf{PA}^\dagger$ as true.

And for a final example, take (4) the theory $\mathsf{Q}^\dagger$ which we get by adding to Q the *negation* of $\forall\mathsf{x}(\mathsf{0} + \mathsf{x} = \mathsf{x})$ as an additional axiom. Like $\mathsf{PA}^\dagger$, $\mathsf{Q}^\dagger$ is also unsound.

But in Section 8.4 we described a re-interpretation of Q's language in which we took the domain of quantification now to be the numbers plus a couple of rogue elements, and then we redefined the successor, addition and multiplication functions for this augmented domain. We can easily see that $\mathsf{Q}^\dagger$'s axioms and hence all its theorems are true on this reinterpretation, and it follows that $\mathsf{Q}^\dagger$ can't prove any contradiction (for contradictions would still be false on the new interpretation). In this case, therefore, we have an interpreted theory which speaks (falsely) about one structure; but we prove it consistent by reinterpreting it as if it described another structure. And on that ground we'll accept $\mathsf{Q}^\dagger$'s canonical Gödel sentence as true.[4]

Why note our different examples (1) to (4)? Two reasons. First, we want to drive home the message that, given a particular canonical Gödel sentence G_T, we might have *various* kinds of grounds for believing it true. But second, our examples also reveal that while our grounds for accepting Gödel sentences may be various, they are – so to speak – perfectly ordinary mathematical reasons. When we met the idea of incompleteness at the very outset, we speculated whether we must have some special, rule-transcending, cognitive grasp of the numbers underlying our ability to recognize (canonical) Gödel sentences as correct arithmetical propositions (see Section 1.4). That speculation should now perhaps begin to seem unnecessarily fanciful.

18.4 What doesn't follow from incompleteness

What follows from our incompleteness theorems so far? In this section, we'll highlight a few claims that someone might carelessly think *are* justified by the theorems, but which certainly aren't.

(a) *'For any nice theory T, T can't prove its Gödel sentences but we always can.'*[5] Not so. Suppose G_T is a canonical Gödel-sentence for theory T. Then, true enough, we can show that, *if* T is indeed nice, then $T \nvdash \mathsf{G}_T$. And hence, *if* we can assume that T is nice, we can conclude that G_T is true.

But this line of thought, of course, doesn't give us a truth-establishing *proof* of G_T, unless we *do* know that T is nice: and that requires in particular knowing that

[4] It is worth commenting on the contrast between the cases of $\mathsf{PA}^\dagger$ and $\mathsf{Q}^\dagger$. We have a proof that $\mathsf{PA}^\dagger$ is consistent if PA is, and we believe that PA *is* consistent: so, because it is a standard metatheorem of first-order logic that any consistent first-order theory has a model (i.e. an interpretation that makes all its theorems true – see Section 22.5), we will accept that there must exist a model of $\mathsf{PA}^\dagger$. So in this case, at least in the first instance, we believe that there *is* such a model because we believe that the theory is consistent (it is quite a challenge to describe a suitable model). In the case of $\mathsf{Q}^\dagger$ the inference goes exactly the other way about, from the existence of a model to the consistency of the theory – and hence to the truth of its Gödel sentence.

[5] This is the thought that underlies the related claim that, roughly, I can't be a machine because (i) no machine M can prove its own Gödel sentence $\mathsf{G}_{T(M)}$ – i.e. the Gödel sentence of the theory $T(M)$ whose theorems are M's potential arithmetical output – but (ii) I can always show that $\mathsf{G}_{T(M)}$ is true. See Section 28.6 for more on this line of thought.

T is consistent.[6] But very often, we *won't* be able to show that T is consistent. And if we don't get a handle on some way of establishing T's consistency, we can't use the proof that, if T is nice, then G_T is true, to show that G_T *is* true.

Take the theory Q_C, which is Q augmented by some unproven Π_1 conjecture C. By Theorem 9.4, to show Q_C is consistent is equivalent to proving C, which we may have no clue how to do. So we may have no clue at all whether Q_C's canonical Gödel sentence is true, even if in fact it is.

(b) *'There are truths of L_A – Gödel sentences – which are not provable in any (nice) axiomatized theory.'* But again, plainly not so. Take any arithmetical truth φ at all. Then this is trivially a theorem of the corresponding formal theory $\mathsf{Q} + \varphi$, i.e. of the theory we get by adding φ as a new axiom to Q. (And, assuming Q is sound, $\mathsf{Q} + \varphi$ is also sound: hence $\mathsf{Q} + \varphi$ is a nice theory – for the theory is consistent because sound, p.r. axiomatized because it still has a finite list of axioms, and is p.r. adequate because it contains Q.)

There is a quantifier-shift fallacy lurking hereabouts: from the theorem that for every nice T there is a truth φ such that $T \nvdash \varphi$ it doesn't follow that there is a truth φ such that for every nice T, $T \nvdash \varphi$.

Of course, deriving a truth φ in some ad hoc formal theory in which we've adopted φ as an axiom is of no interest: what we want are theories which we have independent reasons to accept as sound. So let's now bear that in mind, alongside the observation that we can't always ourselves prove the Gödel sentences for consistent theories. You might wonder if Gödel's First Theorem still gives us reason to accept something along the following lines:

(c) *'For some nice theories T, we can prove their Gödel sentences are true, even though these truths can't be derived in a formal theory we independently accept as sound.'* But once again not so. Take a case where we *can* show that G_T is true, by arguing outside the theory T that T is consistent. Then there's nothing at all in the First Theorem to stop us reflecting on the principles involved in this reasoning, and then constructing a richer formalized theory S (distinct from T, of course) which encapsulates these principles, a formal theory in which we prove G_T to be true. And we will presumably have good reason to accept *this* new formal theory S, since by hypothesis it just regiments some aspects of how we reason anyway.[7]

[6] It is worth mentioning a famous historical episode here: towards the end of the first edition of his *Mathematical Logic* (1940), W. V. Quine proves Gödel's theorem for his proposed formal theory. Should we then conclude that the relevant Gödel sentence is true? We go wrong if we do: for it turns out that Quine's theory in that edition is inconsistent! Every sentence is provable so the system's Gödel sentence which 'says' it is itself unprovable is false.

[7] Careful! All that is being claimed is that there is nothing in the First Theorem that would stop us regimenting the reasoning we use to show G_T is true – when we can show that – into some axiomatized system S. It doesn't follow, of course, that there could be some single master system S^* which wraps up any reasoning we might use in proving true any G_T we can prove, a master system which is itself axiomatizable in a way we can survey. That would be another quantifier-shift fallacy.

In sum, the claims (i), (ii) and (iii) are each fairly obviously wrong. Understand *why* they are wrong, and you will be inoculated against many of the wilder claims about Gödel's First Theorem which are based on such misconceptions.[8]

18.5 What does follow from incompleteness?

So what *can* we infer from the theorem that if a theory is sound/nice, it can't be complete? Well, let's stress here just one point about that simple but deep Gödelian fact: it does sabotage, once and for all, any project of trying to show that all basic arithmetical truths can be thought of as deducible, in some standard deductive system, from one unified set of fundamental axioms which can be specified in a tidy, p.r., way. In short, *arithmetical truth isn't provability in some single axiomatizable system.*

The First Theorem therefore sabotages the logicist project in particular, at least in its classic form of aiming to explain how we can deduce *all* arithmetic from a tidy set of principles which are either logical axioms or definitions.[9] Which at first sight seems a major blow; for there surely remains something very deeply attractive about the thought we expressed at the very beginning of the book, i.e. the thought that in exploring the truths of basic arithmetic, we are just exploring the deductive consequences of a limited set of axiomatic truths which we grasp in grasping the very ideas of the natural number series and the operations of addition and multiplication.

On the other hand, if the truths of even basic arithmetic run beyond what is provable in any given formal system, then even arithmetic is – so to speak – *inexhaustible*. Given any nice theory of arithmetic T we accept as sound, we have to recognize that there are more truths that it cannot prove (there's G_T for a start). Mathematicians are not going to run out even of arithmetical work, as they develop ever richer formal settings in which to prove more truths. Which is, perhaps, a rather more pleasing thought.

But that's enough generalities for the moment: we need to move on. The next three chapters explore more of the technicalities concerning the First Theorem and its close relatives.

[8] We haven't space to explode all the daft misconceptions about the Theorem: see Franzén (2005) for a more wide-ranging demolition job. And also see our Section 28.6.

[9] For the philosophical prospects of non-classical forms of logicism, however, see e.g. Wright (1983), the essays on Frege in Boolos (1998), and Hale and Wright (2001).

19 Strengthening the First Theorem

We now start exploring further around and about our incompleteness theorems. But do be careful! Don't let the developments over these next three chapters obscure the relatively simple lines of the classic Gödelian arguments which we have already given in Chapters 16 and 17.

The main business of this chapter is to present two key ways of strengthening our incompleteness theorems.

1. We show how to extend the reach of both the semantic and syntactic versions of our incompleteness results, so that they apply not just to p.r. axiomatized theories, but to *any* formal theory that counts as axiomatized in the intuitive sense. (We also draw an easy corollary of our newly extended semantic theorem, and prove that the truths of basic arithmetic can't be axiomatized.)

2. We explain how we can do away with the assumption of ω-consistency (as used in Gödel's original First Theorem): we can prove that *any* nice theory is incomplete and incompletable, whether it is ω-consistent or not. That's the Gödel-Rosser Theorem.

Then, after the main business, we explain another way of weakening the assumption of ω-consistency, this time involving the idea of so-called '1-consistency'.

19.1 Broadening the scope of the incompleteness theorems

Our intuitive characterization of a properly formalized theory T requires various properties like that of being an axiom of T to be effectively decidable. Or, what comes to the same given a sensible Gödel numbering scheme, the characteristic functions of numerical properties like that of numbering a T-axiom should be effectively computable (see Sections 3.3 and 11.6). But we now know that not all computable functions are p.r. (Section 11.5). Hence we could in principle have a formal axiomatized theory which isn't *p.r.* axiomatized (in the sense of Section 17.1). Does this give us wriggle room to get around the Gödelian incompleteness theorems in the last chapter? Could there e.g. be a consistent formalized theory of arithmetic containing Q which was complete because not p.r. axiomatized?

Well, as we noted at the start of Section 18.1, a theory T that is formally axiomatized but not p.r. axiomatized will be a rather peculiar beast: checking

that a putative T-proof *is* a proof will then have to involve a non-p.r. open-ended search, which will make T *very* unlike any familiar kind of axiomatized theory. Still, you might say, an oddly axiomatized theory which is complete would be better than no complete formal theory at all. However, we can't get even that.

For consider the following result, which we'll prove in just a moment:

> **Theorem 19.1** *If T is an axiomatized theory, then there is a p.r. axiomatized theory T' which has exactly the same theorems.*

This is an informal version of what's called Craig's Re-axiomatization Theorem.[1]

And now suppose that T is any sound axiomatized theory whose language includes L_A. Craig's Theorem tells us that there is a p.r. axiomatized theory T' which has the same theorems as T. But since T' shares the same theorems, this theory is sound too, so Theorem 17.1 applies. There is therefore an L_A-sentence φ of Goldbach type such that $T' \nvdash \varphi$ and $T' \nvdash \neg\varphi$. Hence, again since T and T' share their theorems,

> **Theorem 19.2** *If a theory T, whose language includes L_A, is axiomatized and is sound, then there is an L_A-sentence φ of Goldbach type such that $T \nvdash \varphi$ and $T \nvdash \neg\varphi$.*

Suppose likewise that T is any consistent axiomatized theory which extends Q. Then Craig's Theorem tells us that there is a p.r. axiomatized theory T' which has the same theorems, so is still consistent, still extends Q, and so is nice (and also T' will be ω-consistent if and only if T is). So Theorem 17.2 applies, and there is an L_A-sentence φ of Goldbach type such that $T' \nvdash \varphi$ and (assuming T' is ω-consistent) $T' \nvdash \neg\varphi$. Hence, since T and T' share their theorems,

> **Theorem 19.3** *If T is a consistent axiomatized theory which extends Q, then there is an L_A-sentence φ of Goldbach type such that $T \nvdash \varphi$ and (assuming T is ω-consistent) $T \nvdash \neg\varphi$.*

In summary: *Given Craig's Theorem, Gödelian incompleteness infects any suitable sound/nice axiomatized theory, whether it is p.r. axiomatized or not.* So it just remains to explain why Craig's Theorem holds:

Proof sketch If T is a formalized theory, its theorems can be effectively enumerated, by Theorem 3.1 (Section 3.5). So imagine stepping through some algorithmic procedure Π which effectively lists T's theorems as $\varphi_0, \varphi_1, \varphi_2, \ldots$. We'll now count the computational steps as we go along, executing one minimal step of the procedure at a time. Most of these steps are interim computations; but very occasionally, the next theorem on the list will be printed out. Suppose that the theorem φ_j is produced at step $s(j)$ as we go through procedure Π.

[1] Equivalently, it says that if a body of wffs is axiomatizable at all, then we can find a p.r. axiomatization of it – where, amplifying the definition of Section 5.4, a set of wffs Γ is (p.r.) axiomatizable if there is a (p.r.) axiomatized formal theory T such that, for any wff φ, $\varphi \in \Gamma$ if and only if $T \vdash \varphi$. For the original theorem, see Craig (1953).

Now for Craig's ingenious trick. Consider the derivative theory T' defined as follows: (i) for each T-theorem φ_j, T' has the axiom $(\varphi_j \wedge \varphi_j \wedge \varphi_j \wedge \ldots \wedge \varphi_j)$, where this is an n-fold conjunction with $n = s(j)$ conjuncts; (ii) T''s sole rule of inference is $\wedge$-elimination.

Trivially, every T-theorem is a T'-theorem. And equally trivially, making the sole assumption that T has the usual rule of $\wedge$-introduction, every T'-theorem is a T-theorem. (Exercise: tweak the definition of T' so that we no longer need even to assume that T has $\wedge$-introduction.)

Given an arbitrary wff ψ, here's how to tell whether it is a T'-axiom:

1. Read in ψ and see if it is the n-fold conjunction of some wff φ (that can be done with a search bounded by the length of ψ, i.e. by routines using 'for' loops). If it passes that test, count the number of conjuncts n, and then move on to ...

2. Run through n steps of the procedure Π which lists off the theorems of T (think of that as executing one big 'for' loop counting from 1 to n).

3. Check to see if at the end of these n steps, Π prints out the same wff φ (you can do that with another 'for' loop).

Evidently, if the final check confirms that φ is indeed printed out, its n-fold conjunction ψ is a T'-axiom; otherwise ψ isn't an axiom. Which shows, therefore, that testing whether ψ is a T'-axiom can be done by a computational procedure which uses 'for' loops, without open-ended searches. Hence the characteristic function of the property of numbering a T'-axiom will be primitive recursive. And therefore T' is p.r. axiomatized. ⊠

Here's an immediate corollary worth noting:

> **Theorem 19.4** *If Σ is a set of wffs which can be effectively enumerated, then there is a p.r. axiomatized theory T' whose theorems are exactly the members of Σ.*

Proof As before, starting from the second line! ⊠

To be sure, that's a bit quick-and-dirty. However we won't pause to tidy things up now. That's because in Chapter 30 – when we at last have a general account of computation and decidability to hand – we'll be returning again to our more encompassing Theorems 19.2 and 19.3, and we will then be able to prove them *without* going via Craig's Theorem. But still, even before we go into that more general account of computation, it has been well worth noting that there is an intuitively compelling argument that our Gödelian arguments which officially apply just to p.r. axiomatized theories in fact apply more generally.

19.2 True Basic Arithmetic can't be axiomatized

We now extract an immediate corollary of Theorem 19.2. Let True Basic Arithmetic be $\mathcal{T}_A$, the set of truths of L_A, i.e. the set of sentences which are true on the standard interpretation built into L_A. Then we have

Theorem 19.5 *$\mathcal{T}_A$, True Basic Arithmetic, is not axiomatizable.*

Proof Suppose otherwise, i.e. suppose T is an axiomatized theory which proves exactly the truths in $\mathcal{T}_A$. Then T's theorems are all true, so T is sound; its language includes L_A; so Theorem 19.2 applies. Hence there is an L_A-sentence φ such that $T \nvdash \varphi$ and $T \nvdash \neg\varphi$. But one of φ and $\neg\varphi$ must be true, which means that T doesn't prove all the truths in $\mathcal{T}_A$. Contradiction. ⊠

Compare this result with Theorem 5.6, which tells us the truths of a 'sufficiently expressive' language are not axiomatizable. It is plain what we have gained. The old theorem didn't tell us anything about what a sufficiently expressive language is like; it left open the possibility that the theorem only applied to *very* rich languages. Now we see that non-axiomatizability applies even down at the lowly level of the truths of L_A.

19.3 Rosser's improvement

(a) Moving on to our second main topic of the chapter, recall that one half of the First Theorem requires the assumption that we are dealing with a theory T which is not only nice but is ω-consistent. But we can improve on this in two different ways:

1. We can keep the *same* undecidable sentence G_T while invoking the weaker assumption of '1-consistency' in showing that $T \nvdash \neg\mathsf{G}_T$.

2. Following Barkley Rosser (1936), we can construct a *different* and more complex sentence R_T such that we only need to assume T is plain consistent in order to show that R_T is formally undecidable.

Since Rosser's clever construction yields the better result, we'll start with that.

(b) How does Rosser construct an undecidable sentence R_T for T? Well, essentially, where Gödel constructs a sentence G_T that indirectly says 'I am unprovable in T', Rosser constructs a sentence R_T which indirectly says 'if I am provable in T, then my negation is already provable' (i.e. it says that if there is a proof of R_T with super g.n. n, then there is a proof of $\neg\mathsf{R}_T$ with a smaller code number).

This sentence turns out to be Π_1, And another semantic incompleteness result is immediate: if T is a nice *sound* theory (and hence consistent), neither R_T nor $\neg\mathsf{R}_T$ can be provable:

Proof Assume T soundness. And suppose R_T were a theorem. Then it would be true. In other words, 'if R_T is provable, $\neg\mathsf{R}_T$ is already provable' would be true,

and also this conditional would have a true antecedent. So we can infer that $\neg\mathsf{R}_T$ is provable. Which makes T inconsistent, contrary to hypothesis. Therefore R_T is unprovable. Which shows that the material conditional 'if R_T is provable, $\neg\mathsf{R}_T$ is already provable' has a false antecedent, and hence is true. In other words, R_T is true. Hence its negation $\neg\mathsf{R}_T$ is false, and is therefore unprovable. ⊠

(c) As we said, however, in order to show that neither R_T nor $\neg\mathsf{R}_T$ is provable we don't need the semantic assumption that T is sound. *The syntactic assumption of T's consistency is enough.* So we get the *Gödel-Rosser Theorem*:

> **Theorem 19.6** *If T is a nice theory, then there is an L_A-sentence φ of Goldbach type such that neither $T \vdash \varphi$ nor $T \vdash \neg\varphi$.*

Up to now, then, the two halves of our syntactic incompleteness theorem have been asymmetric: we used the weak condition of consistency in one half, and then a stronger condition of ω-consistency in the other half. Rosser restores symmetry.

However, we'll hang fire just for the moment on spelling out how to find a Rosser sentence and so give a proof of the pivotal Theorem 19.6. For things go perhaps just a little bit more cleanly if we wait until after we've met the Diagonalization Lemma in the next chapter (the proof therefore comes in Section 21.3).[2] Be patient!

19.4 1-consistency and Σ_1-soundness

(a) Rosser improves on Gödel's original Theorem by replacing the neatly defined G_T by a more complex undecidable sentence R_T; the ensuing argument that R_T is undecidable, assuming T is consistent, is correspondingly more intricate. But two decades after Rosser, Georg Kreisel (1957) noted that we can in fact keep Gödel's own argumentation for the First Theorem in place, while weakening the assumption of ω-consistency to *1-consistency* or equivalently Σ_1-*soundness*.

Since those twin notions involve something stronger than mere consistency, Kreisel's observation doesn't give us as good a result as Rosser's. Still, we'd better pause to explain what's going on, if only because so many modern presentations of Gödel's First Theorem do in fact state it in terms of 1-consistency and/or Σ_1-soundness. (You are, however, *very* welcome to skip the following slightly fiddly discussion.)

(b) If you are still reading, then we'll begin with a reminder and two new definitions:

i. T is *ω-consistent* if there is no $\varphi(\mathsf{x})$ such that $T \vdash \exists\mathsf{x}\varphi(\mathsf{x})$ while, for each m, $T \vdash \neg\varphi(\overline{\mathsf{m}})$.

[2] Enthusiasts who want to scale the Gödel-Rosser Theorem by the direct route can consult e.g. Mendelson (1964, pp. 145–146), or the slightly less tractable Mendelson (1997, pp. 209–210).

ii. T is *1-consistent* iff there is no Δ_0 wff $\varphi(\vec{x})$ such that $T \vdash \exists\vec{x}\varphi(\vec{x})$ while, for each $\vec{m}$, $T \vdash \neg\varphi(\vec{m})$.[3]

iii. T is Σ_1-*sound* iff, for any Σ_1 sentence φ such that $T \vdash \varphi$, φ is true. (Cf. Section 9.7)

(c) Now take another quick look at our demonstration in Section 16.7 that, if PA is ω-consistent, it can't prove $\neg$G. Assume that PA *is* ω-consistent. We then noted that if PA *did* prove G, we'd have both (1) $\mathsf{PA} \vdash \exists x\mathsf{Gdl}(x, \ulcorner\mathsf{U}\urcorner)$ yet also (2) $\mathsf{PA} \vdash \neg\mathsf{Gdl}(m, \ulcorner\mathsf{U}\urcorner)$ for each m, contradicting the assumption of ω-consistency. A little reflection, however, shows that we don't need to appeal to ω-consistency to get a contradiction here: if PA is 1-consistent, it can't prove $\neg$G.

Proof Remember that $\mathsf{Gdl}(x, \ulcorner\mathsf{U}\urcorner)$ is Σ_1: so it is equivalent to something of the form $\exists\vec{z}\psi(x, \vec{z})$, where ψ is Δ_0. So the conditions (1) and (2) come to this, where ψ is Δ_0: (1) $\mathsf{PA} \vdash \exists x\exists\vec{z}\psi(x, \vec{z})$ and (2) $\mathsf{PA} \vdash \neg\exists\vec{z}\psi(m, \vec{z})$ for each m. And (2) elementarily implies (3) $\mathsf{PA} \vdash \neg\psi(m, \vec{n})$ for every sequence of numbers $\vec{n}$.

So suppose PA is 1-consistent. Then you can't have (1) and (3); so you can't have (1) and (2); so $\mathsf{PA} \nvdash \neg$G. ⊠

(d) Here are two more very easy lemmas:

1. If T is Σ_1-sound it is consistent.

2. If T is nice, T is Σ_1-sound if and only if it is 1-consistent.

Proof for (1) If T is inconsistent, we can derive anything in T – including e.g. $0 = 0$ and $0 \neq 0$, which are both trivially Σ_1. But those two sentences can't both be true. Hence, if T is inconsistent, it can't be Σ_1-sound. Contraposing, Σ_1-soundness implies consistency. ⊠

Proof for (2) Suppose that T is (i) Σ_1-sound but (ii) 1-inconsistent. Then by (ii) there is a Δ_0 wff $\varphi(\vec{x})$ such that $T \vdash \exists\vec{x}\varphi(\vec{x})$, but for each $\vec{m}$, $T \vdash \neg\varphi(\vec{m})$. But by (i), if $T \vdash \exists\vec{x}\varphi(\vec{x})$, then $\exists\vec{x}\varphi(\vec{x})$ is true. And so for some $\vec{m}$, $\varphi(\vec{m})$ must be true. But being Δ_0, that will be provable in T (since T contains Q, and Q correctly decides every Δ_0 wff). So for some $\vec{m}$, $T \vdash \varphi(\vec{m})$. So T is inconsistent, contradicting the assumption of Σ_1-soundness. So if (i), then not-(ii), which gives us one direction of the biconditional.

For the other direction (contraposed), suppose T is *not* Σ_1-sound, i.e. proves some false strictly Σ_1 wff. So for some Δ_0 wff φ, $T \vdash \exists\vec{x}\varphi(\vec{x})$ yet $\exists\vec{x}\varphi(\vec{x})$ is false.

[3] Or at least, let that be our official definition – where, of course, $\vec{x}$ is a tuple of variables, $\exists\vec{x}$ is a bunch of quantifiers, one for each variable, and $\vec{m}$ is a tuple of numbers of matching length.

Equally often, 1-consistency is defined in terms of there being no Δ_0 wff $\varphi(x)$ such that $T \vdash \exists x\varphi(x)$ while, for each m, $T \vdash \neg\varphi(m)$. In virtue of Theorem 10.1, this one-quantifier version is equivalent to our official multi-quantifier version in the presence of a smidgin of induction. And the one-quantifier version does have the nice feature of being transparently a weaker condition than ω-consistency.

So each instance of $\neg\varphi(\vec{\mathsf{m}})$ will be true and – being Δ_0 – will be provable in T. Then T is 1-inconsistent. ⊠

(e) Generalizing the argument in (c) and putting everything together, we therefore get the following revised versions of the First Theorem, which (unlike the stronger Gödel-Rosser Theorem) are still proved essentially via Gödel's original construction:

> **Theorem 19.7** *If T is a nice Σ_1-sound theory, then there is an L_A sentence φ of Goldbach type which is formally undecidable by T.*
>
> **Theorem 19.8** *If T is a nice 1-consistent theory, then there is an L_A sentence φ of Goldbach type which is formally undecidable by T.*

If we want to stress the naturalness of the weak condition which suffices for proving incompleteness via Gödel's construction, then we might want to emphasize Σ_1-soundness. If we want to stress that the condition in question can in fact be characterized entirely syntactically, without using the notion of truth, then we'll emphasize 1-consistency instead.

20 The Diagonalization Lemma

In this short but pivotal chapter, we now show how a version of Gödel's trick in constructing G_T can be generalized to give us a proof of the so-called Diagonalization Lemma. This Lemma will then be used in the next chapter to prove some key theorems, including the Gödel-Rosser Theorem and Tarski's Theorem.

So, in this chapter,

1. We first introduce the provability predicate Prov_T which applies to a number if it numbers a T-theorem ...
2. ... and we prove a simple theorem about provability predicates.
3. We then use Prov_T in constructing a wff equivalent to T's canonical Gödel sentence G_T.
4. That leads to a quick proof that if T is nice, $T \vdash \mathsf{G}_T \leftrightarrow \neg\mathsf{Prov}_T(\ulcorner \mathsf{G}_T \urcorner)$. Which means that not only is T's canonical Gödel sentence true if and only if it is unprovable, but we can derive that claim inside T itself.
5. Put $\varphi(\mathsf{x}) =_{\text{def}} \neg\mathsf{Prov}_T(\mathsf{x})$. We have just said that if T is a nice theory, then there is a sentence γ such that $T \vdash \gamma \leftrightarrow \varphi(\ulcorner \gamma \urcorner)$. We now generalize this observation to get the full Diagonalization Lemma. Let $\varphi(\mathsf{x})$ be *any* open wff with one free variable: it will *always* be the case that, if T is nice, there is some sentence γ such that $T \vdash \gamma \leftrightarrow \varphi(\ulcorner \gamma \urcorner)$.

20.1 Provability predicates

Recall: $Prf(m, n)$ holds when m is the super g.n. for a PA-proof of the sentence with g.n. n. So $Prov(n) =_{\text{def}} \exists v\, Prf(v, n)$ will hold when some number codes a proof of the sentence with g.n. n – i.e. when n numbers a PA-theorem. (See Sections 15.2, 15.3, and also the very end of Section 15.9.)

Since the relation Prf is p.r., we know it can be captured in PA by a Σ_1 wff that perspicuously recapitulates the p.r. definition of Prf (see Section 13.8). So let's adopt the following notational convention:

> $\mathsf{Prf}(\mathsf{x}, \mathsf{y})$ stands in for a Σ_1 wff which canonically captures Prf.

And now here's another definition. We put

> $\mathsf{Prov}(\mathsf{x}) =_{\text{def}} \exists \mathsf{v}\, \mathsf{Prf}(\mathsf{v}, \mathsf{x})$.[1]

[1] We will assume, as we can without loss of generality, that we can quantify using 'v' without clash of variables.

Evidently, this wff is also Σ_1; and it is true of n just when n numbers a PA-theorem. We'll call $\mathsf{Prov}(\mathsf{x})$ a *(canonical) provability predicate* for PA. It evidently *expresses* the provability property *Prov*. But be careful! – we mustn't assume that it *captures* that property: in fact, we'll soon prove that it doesn't (see Theorem 21.3).

Next, we generalize in the now familiar way. For any theory T, there is similarly a relation $Prf_T(m, n)$ which holds when m is the super g.n. for a T-proof of the sentence with g.n. n.[2] If T is a nice theory, then Prf_T will again be a p.r. relation. Hence there will be a corresponding Σ_1 wff $\mathsf{Prf}_T(\mathsf{x}, \mathsf{y})$ which captures this relation in T (and again captures it in a perspicuous way, by recapitulating the p.r. definition of Prf_T). So we can define $\mathsf{Prov}_T(\mathsf{x}) =_{\text{def}} \exists \mathsf{v}\, \mathsf{Prf}_T(\mathsf{v}, \mathsf{x})$, where this new provability predicate expresses the property of numbering a T-theorem.

20.2 An easy theorem about provability predicates

(a) Here's a straightforward result about provability predicates:

Theorem 20.1 *Let T be a nice theory. Then for any sentence φ:*

C1. If $T \vdash \varphi$, then $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$;
Cω. Suppose T is ω-consistent: then if $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$, $T \vdash \varphi$.

Proof for (C1) First assume $T \vdash \varphi$. Then there is a T proof of the wff with g.n. $\ulcorner\varphi\urcorner$. Let this proof have the super g.n. m. Then, by definition, $Prf_T(m, \ulcorner\varphi\urcorner)$. Hence, since Prf_T is captured by Prf_T, it follows that $T \vdash \mathsf{Prf}_T(\overline{\mathsf{m}}, \ulcorner\varphi\urcorner)$. Hence $T \vdash \exists \mathsf{v}\, \mathsf{Prf}_T(\mathsf{v}, \ulcorner\varphi\urcorner)$, i.e. $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$. ⊠

An even quicker proof for (C1) If $T \vdash \varphi$, then $\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$ will be true. But $\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$ is Σ_1; hence, since Q is Σ_1 complete, i.e. proves all true Σ_1 sentences (by Theorem 9.2), $\mathsf{Q} \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$. Hence $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$. ⊠

Proof for (Cω) Now assume T is ω-consistent and also that $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$, i.e. $T \vdash \exists \mathsf{v}\, \mathsf{Prf}_T(\mathsf{v}, \ulcorner\varphi\urcorner)$. Suppose, for reductio, that $T \nvdash \varphi$. Then, for all m, it isn't the case that $Prf_T(m, \ulcorner\varphi\urcorner)$. Therefore, since T is nice and captures Prf_T, for all m, $T \vdash \neg\mathsf{Prf}_T(\overline{\mathsf{m}}, \ulcorner\varphi\urcorner)$. But that makes T ω-inconsistent, contrary to hypothesis.[3] ⊠

(b) Two quick comments on our easy theorem. First, suppose that we are given only that T is nice, this time *without* the further assumption of something like

[2] Of course, we are now talking in the context of some appropriate scheme for Gödel-numbering expressions of T – see the second remark towards the end of Section 15.1.

[3] If you delved into Section 19.4, then you will guess that the condition of ω-consistency in our theorem can be weakened to 1-consistency or to Σ_1-soundness. And you'd be right. It is just trivial that if T is Σ_1-sound and $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$, then – since Prov_T *is* Σ_1 – $\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$ is true, i.e. $T \vdash \varphi$.

ω-consistency. Then it *won't* always be the case that if $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$ then $T \vdash \varphi$. Why? Roughly: if T is ω-inconsistent, then T is not sound, i.e. it has false theorems on the standard interpretation (see Section 16.6 (b)). But if T is not sound, then among the things that T can get wrong are facts about what it can prove.

Second, with the additional assumption of ω-consistency back in place, it follows from (Cω) that if $T \nvdash \varphi$ then $T \nvdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$. But this fact needs to be *very* sharply distinguished from the claim that, if $T \nvdash \varphi$, then $T \vdash \neg\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$. *That* second claim is in fact plain *false*.

For suppose otherwise. Then this supposition, combined with (C1), implies that Prov_T *captures* the provability property $Prov_T$. However, as we'll soon see in Section 21.4, *no* wff can do that if T is nice. So, even in nice theories, it isn't always true that if $T \nvdash \varphi$, then $T \vdash \neg\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$. But there's more: in Section 25.2 (d), we'll see that in typical nice theories, $T \vdash \neg\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$ is *never* true. To put it vividly, such theories may know about what they *can* prove, but they know *nothing* about what they *can't* prove!

20.3 G and Prov

We saw that our Gödel sentence G for PA is constructed in such a way that it is true if and only if it is unprovable. Using the provability predicate Prov, we can now *express* this fact about G inside PA by the sentence

$$\mathsf{G} \leftrightarrow \neg\mathsf{Prov}(\ulcorner\mathsf{G}\urcorner).$$

But we can do better: we can actually *prove* this very sentence inside PA.

To show this, our next target theorem, let's start by thinking again about how our Gödel sentence for PA was constructed in Section 16.2. We'll then do some manipulation to find a sentence which is provably equivalent to G and which explicitly involves Prov. So, recall,

$$\mathsf{G} =_{\text{def}} \exists\mathsf{y}(\mathsf{y} = \ulcorner\mathsf{U}\urcorner \wedge \mathsf{U}).$$

Here, '$\ulcorner\mathsf{U}\urcorner$' stands in for the numeral for U's g.n., and

$$\mathsf{U} =_{\text{def}} \forall\mathsf{x}\neg\mathsf{Gdl}(\mathsf{x},\mathsf{y})$$

where $\mathsf{Gdl}(\mathsf{x},\mathsf{y})$ captures our old friend, the relation Gdl.

Now, by definition,

$$Gdl(m,n) =_{\text{def}} Prf(m, diag(n)).$$

But the one-place p.r. function $diag$ can be captured as a function in PA by some open wff $\mathsf{Diag}(\mathsf{x},\mathsf{y})$. We can therefore give the following definition:

$$\mathsf{Gdl}(\mathsf{x},\mathsf{y}) =_{\text{def}} \exists\mathsf{z}(\mathsf{Prf}(\mathsf{x},\mathsf{z}) \wedge \mathsf{Diag}(\mathsf{y},\mathsf{z})).$$

And now let's do some elementary manipulations:

$$
\begin{array}{lll}
\mathsf{U} =_{\text{def}} \forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \mathsf{y}) & & \\
\quad =_{\text{def}} \forall \mathsf{x} \neg \exists \mathsf{z}(\mathsf{Prf}(\mathsf{x}, \mathsf{z}) \wedge \mathsf{Diag}(\mathsf{y}, \mathsf{z})) & & \text{(definition of Gdl)} \\
\quad \leftrightarrow \quad \forall \mathsf{x} \forall \mathsf{z} \neg (\mathsf{Prf}(\mathsf{x}, \mathsf{z}) \wedge \mathsf{Diag}(\mathsf{y}, \mathsf{z})) & & \text{(pushing in the negation)} \\
\quad \leftrightarrow \quad \forall \mathsf{z} \forall \mathsf{x} \neg (\mathsf{Prf}(\mathsf{x}, \mathsf{z}) \wedge \mathsf{Diag}(\mathsf{y}, \mathsf{z})) & & \text{(swapping quantifiers)} \\
\quad \leftrightarrow \quad \forall \mathsf{z}(\mathsf{Diag}(\mathsf{y}, \mathsf{z}) \rightarrow \neg \exists \mathsf{x}\, \mathsf{Prf}(\mathsf{x}, \mathsf{z})) & & \text{(rearranging after '}\forall \mathsf{z}\text{')} \\
\quad \leftrightarrow \quad \forall \mathsf{z}(\mathsf{Diag}(\mathsf{y}, \mathsf{z}) \rightarrow \neg \exists \mathsf{v}\, \mathsf{Prf}(\mathsf{v}, \mathsf{z})) & & \text{(changing variables)} \\
\quad =_{\text{def}} \forall \mathsf{z}(\mathsf{Diag}(\mathsf{y}, \mathsf{z}) \rightarrow \neg \mathsf{Prov}(\mathsf{z})) & & \text{(definition of Prov)} \\
\quad =_{\text{def}} \mathsf{U}' & & \text{(new abbreviation)}
\end{array}
$$

Since the U/U' equivalence is proved by simple logical manipulations, that means we can prove the equivalence inside the formal first-order logic built into PA.

A quick comment. Since U and U' are trivially equivalent, it can't matter whether we work with G, the diagonalization of U, or G', the diagonalization of U'. These are distinct wffs involving different numerals introduced when we do the diagonalization: but it is an easy exercise to check that G' will do just as well as G for proving Gödel's theorem. We won't pause over this.

20.4 Proving that G is equivalent to $\neg\mathsf{Prov}(\ulcorner \mathsf{G} \urcorner)$

To take us up to the theorem we want, note that first-order logic can of course prove the trivial equivalence of $\mathsf{G} \leftrightarrow \mathsf{U}(\ulcorner \mathsf{U} \urcorner)$; and we've just said that it can also prove the U/U' equivalence. So it can prove $\mathsf{G} \leftrightarrow \mathsf{U}'(\ulcorner \mathsf{U} \urcorner)$. Since PA includes the necessary first-order logic we therefore get, unpacking just a bit,

$$\mathsf{PA} \vdash \mathsf{G} \leftrightarrow \forall \mathsf{z}(\mathsf{Diag}(\ulcorner \mathsf{U} \urcorner, \mathsf{z}) \rightarrow \neg \mathsf{Prov}(\mathsf{z})).$$

Now, diagonalizing U yields G. Hence, just by the definition of *diag*, we have $diag(\ulcorner \mathsf{U} \urcorner) = \ulcorner \mathsf{G} \urcorner$. Since by hypothesis Diag captures *diag* as a function, it follows from the definition in Section 12.2 that

$$\mathsf{PA} \vdash \forall \mathsf{z}(\mathsf{Diag}(\ulcorner \mathsf{U} \urcorner, \mathsf{z}) \leftrightarrow \mathsf{z} = \ulcorner \mathsf{G} \urcorner).$$

Putting those two results together, we immediately get

$$\mathsf{PA} \vdash \mathsf{G} \leftrightarrow \forall \mathsf{z}(\mathsf{z} = \ulcorner \mathsf{G} \urcorner \rightarrow \neg \mathsf{Prov}(\mathsf{z})).$$

But the right-hand side of that biconditional is trivially equivalent to $\neg\mathsf{Prov}(\ulcorner \mathsf{G} \urcorner)$. So we've proved our first desired result:

Theorem 20.2 $\mathsf{PA} \vdash \mathsf{G} \leftrightarrow \neg\mathsf{Prov}(\ulcorner \mathsf{G} \urcorner)$.

To repeat, what this shows is that the informal claim 'G is true if and only if it is unprovable' can itself be formally proved within PA. Very neat!

And the same reasoning applies to other theories which contain first-order logic and which can capture p.r. functions and relations. In other words, if Prov_T is the provability predicate for T constructed analogously to the predicate Prov for PA, and if G_T is a Gödel sentence constructed analogously to G, then by exactly the same argument we have

Theorem 20.3 *If T is a nice theory,* $T \vdash \mathsf{G}_T \leftrightarrow \neg\mathsf{Prov}_T(\ulcorner \mathsf{G}_T \urcorner)$.

20.5 Deriving the Lemma

(a) A little more reflection, however, shows that this last result is just an illustration of a quite general *Diagonalization Lemma* which is the main target of this chapter: [4]

> **Theorem 20.4** *If T is a nice theory and $\varphi(\mathsf{x})$ is* any *wff of its language with one free variable, then there is a sentence γ of T's language such that $T \vdash \gamma \leftrightarrow \varphi(\ulcorner\gamma\urcorner)$.*

That's wonderful! The original Gödel construction gives us a sentence that T shows is true if and only if it satisfies the condition of being unprovable-in-T. Now it turns out that it doesn't matter what condition we take, so long as it is appropriately expressible in T's language: there will be a sentence which T shows is true if and if only if it satisfies that condition.

Proof We use the same basic proof idea as before. In other words, we do to the generic 'φ' pretty much what our Gödelian construction above did to '$\neg\mathsf{Prov}$'. So, first step, put $\psi(\mathsf{y}) =_{\text{def}} \forall\mathsf{z}(\mathsf{Diag}_T(\mathsf{y},\mathsf{z}) \to \varphi(\mathsf{z}))$ where Diag_T captures the diagonalization function as a function in T.

Next, of course, we do some diagonalization! So construct γ, the diagonalization of $\psi(\mathsf{y})$. This is trivially equivalent to $\psi(\ulcorner\psi\urcorner)$, and this trivial equivalence can be proved in T. So $T \vdash \gamma \leftrightarrow \forall\mathsf{z}(\mathsf{Diag}_T(\ulcorner\psi\urcorner,\mathsf{z}) \to \varphi(\mathsf{z}))$.

Our theorem now follows very speedily. For note that $diag_T(\ulcorner\psi\urcorner) = \ulcorner\gamma\urcorner$. Hence, $T \vdash \forall\mathsf{z}(\mathsf{Diag}_T(\ulcorner\psi\urcorner,\mathsf{z}) \leftrightarrow \mathsf{z} = \ulcorner\gamma\urcorner)$, since by hypothesis Diag_T captures $diag_T$.

It follows that $T \vdash \forall\mathsf{z}(\mathsf{Diag}_T(\ulcorner\psi\urcorner,\mathsf{z}) \to \varphi(\mathsf{z})) \leftrightarrow \forall\mathsf{z}(\mathsf{z} = \ulcorner\gamma\urcorner \to \varphi(\mathsf{z}))$, since we have just shown that the antecedents of the conditionals on either side are provable equivalents.

But we've already seen that the left-hand side of our biconditional is provably equivalent to γ; and the right-hand side is in turn trivially equivalent to $\varphi(\ulcorner\gamma\urcorner)$. So $T \vdash \gamma \leftrightarrow \varphi(\ulcorner\gamma\urcorner)$. ⊠

Alternative proof Reviewing that first proof, it is quite easy to spot that a variant construction is possible; and it is worth quickly spelling this out. So now put $\psi'(\mathsf{y}) =_{\text{def}} \exists\mathsf{z}(\mathsf{Diag}_T(\mathsf{y},\mathsf{z}) \wedge \varphi(\mathsf{z}))$. Redefine γ to be the diagonalization of $\psi'(\mathsf{y})$. Then $T \vdash \gamma \leftrightarrow \exists\mathsf{z}(\mathsf{Diag}_T(\ulcorner\psi'\urcorner,\mathsf{z}) \wedge \varphi(\mathsf{z}))$.

Since $diag_T(\ulcorner\psi'\urcorner) = \ulcorner\gamma\urcorner$, we have $T \vdash \forall\mathsf{z}(\mathsf{Diag}_T(\ulcorner\psi'\urcorner,\mathsf{z}) \leftrightarrow \mathsf{z} = \ulcorner\gamma\urcorner)$.

Hence $T \vdash \exists\mathsf{z}(\mathsf{Diag}_T(\ulcorner\psi'\urcorner,\mathsf{z}) \wedge \varphi(\mathsf{z})) \leftrightarrow \exists\mathsf{z}(\mathsf{z} = \ulcorner\gamma\urcorner \wedge \varphi(\mathsf{z}))$, since the first conjuncts on either side are equivalents. But the left-hand side of our biconditional is provably equivalent to γ, and the right-hand side is equivalent to $\varphi(\ulcorner\gamma\urcorner)$. So again $T \vdash \gamma \leftrightarrow \varphi(\ulcorner\gamma\urcorner)$. ⊠

[4] This conventional name is apt given the role of diagonalization in its proof. However, 'lemma' usually connotes a minor result or interim result: this Lemma is assuredly more important than that! In a footnote added to later reprintings of Gödel (1934), Gödel says that this Lemma 'was first noted by Carnap (1934)': first noted in print, yes; but it has been suggested that Gödel himself had already got there in 1930.

(b) A quick remark about jargon. Suppose that the function f maps the argument a back to a itself, so that $f(a) = a$: then a is said to be a *fixed point* for f. And a theorem to the effect that, under certain conditions, there is a fixed point for f is a *fixed-point theorem*. By a slightly strained analogy, the Diagonalization Lemma is also standardly referred to as a fixed point theorem, with γ behaving like a 'fixed point' for the predicate $\varphi(\mathsf{x})$.

(c) We'll end the chapter by noting two easy corollaries for later use.

1. *If T is a nice theory, and $\varphi(\mathsf{x})$ is a Π_1 open wff of its language with one free variable, then there is a Π_1 fixed point for $\varphi(\mathsf{x})$.*

2. *If T is nice but an* unsound *theory, and $\varphi(\mathsf{x})$ is any wff of its language with one free variable, then $\varphi(\mathsf{x})$ has a* false *fixed point.*

Proof for (1) Look again at the first proof for the Lemma. Note that the wff Diag_T which captures the p.r. $diag_T$ function in the standard way will be Σ_1. Then it is easy to see that if φ is Π_1, then $\psi(\mathsf{y}) =_{\text{def}} \forall\mathsf{z}(\mathsf{Diag}_T(\mathsf{y},\mathsf{z}) \to \varphi(\mathsf{z}))$ will also be Π_1. Hence the fixed point γ which is equivalent to $\psi(\ulcorner\psi\urcorner)$ is also Π_1. ⊠

Proof for (2) Pick θ to be a false theorem of the theory T (it has some, being unsound). Suppose $\mathsf{Diag}_T(\mathsf{x},\mathsf{y})$ captures $diag_T$. Then so does $\mathsf{Diag}'_T(\mathsf{x},\mathsf{y}) =_{\text{def}} [\mathsf{Diag}_T(\mathsf{x},\mathsf{y}) \wedge \theta]$, by the general point we noted in Section 4.6 (b).

Now our proofs of the Diagonalization Lemma don't depend on which particular wff serves to capture $diag_T$. So Diag'_T will do just as well as Diag_T. In particular the second proof above will go through with the fixed point γ identified as the diagonalization of $\exists\mathsf{z}(\mathsf{Diag}'_T(\mathsf{y},\mathsf{z}) \wedge \varphi(\mathsf{z}))$, i.e. the diagonalization of $\exists\mathsf{z}([\mathsf{Diag}_T(\mathsf{y},\mathsf{z}) \wedge \theta] \wedge \varphi(\mathsf{z}))$.

So γ is of the form $\exists\mathsf{y}(\mathsf{y} = \ulcorner\ldots\urcorner \wedge \exists\mathsf{z}([\mathsf{Diag}_T(\mathsf{y},\mathsf{z}) \wedge \theta] \wedge \varphi(\mathsf{z})))$. Hence γ has θ as an embedded conjunct, and will therefore be false if θ is false. ⊠

21 Using the Diagonalization Lemma

In this chapter, we use the really rather beautiful Diagonalization Lemma a number of times over. Here's a quick guide through the sections:

1. First, it is worth seeing how we can use the Lemma to prove Gödel's First Theorem again. We can think of the Theorem as in fact generated by putting together two separate strands of thought – (i) the general Diagonalization Lemma which tells us that the wff $\neg\mathsf{Prov}(\mathsf{x})$ in particular has a fixed point, plus (ii) reflections on the logical behaviour of $\mathsf{Prov}(\mathsf{x})$.
2. After a brief aside on the very idea of a Gödel sentence ...
3. ... we then use the Diagonalization Lemma again to derive the Gödel-Rosser Theorem which previously we left unproved.
4. Next, we show that no nice theory T can capture its own provability property $Prov_T$.
5. Then we prove Tarski's key Theorem about the 'indefinability' of truth.
6. We note that this gives us what in a sense might be thought of as the master argument for incompleteness.
7. Finally, as a coda, we show some results that concern how *long* the proofs have to be.

What follows is inevitably an action-packed chapter: take it slowly!

21.1 The First Theorem again

Theorem 20.1 established that the following two conditions obtain for provability predicates for nice theories T:

C1. If $T \vdash \varphi$, then $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$;

Cω. If T is ω-consistent then, if $T \vdash \mathsf{Prov}_T(\ulcorner\varphi\urcorner)$, $T \vdash \varphi$.

And these two principles immediately imply

> **Theorem 21.1** *Let T be a nice theory, and let γ be* any *fixed point for $\neg\mathsf{Prov}_T(\mathsf{x})$. Then $T \nvdash \gamma$; and if T is ω-consistent, then $T \nvdash \neg\gamma$.*

Proof For readability, let's start dropping subscript 'T's when context readily supplies them. By hypothesis, $T \vdash \gamma \leftrightarrow \neg\mathsf{Prov}(\ulcorner\gamma\urcorner)$. So if $T \vdash \gamma$ then $T \vdash \neg\mathsf{Prov}(\ulcorner\gamma\urcorner)$. But, by (C1), if $T \vdash \gamma$ then $T \vdash \mathsf{Prov}(\ulcorner\gamma\urcorner)$. So, given T is consistent, we can't have $T \vdash \gamma$.

Now assume T is ω-consistent and hence plain consistent, and suppose $T \vdash \neg\gamma$. Since $T \vdash \gamma \leftrightarrow \neg\mathsf{Prov}(\ulcorner\gamma\urcorner)$, it follows $T \vdash \mathsf{Prov}(\ulcorner\gamma\urcorner)$. But by consistency, if $T \vdash \neg\gamma$, then $T \nvdash \gamma$. Hence by (Cω), $T \nvdash \mathsf{Prov}(\ulcorner\gamma\urcorner)$. Contradiction. So, assuming T *is* ω-consistent, we can't have $T \vdash \neg\gamma$. ⊠

In sum, Theorem 21.1 tells us that if there *is* a fixed point for $\neg\mathsf{Prov}_T$, then T can't be negation complete, assuming it is nice and ω-consistent. But Theorem 20.4 tells us that such a fixed point must exist; and its first corollary tells us that the fixed point will be of Goldbach type since $\neg\mathsf{Prov}_T$ is Π_1. Put the results together and we've got the First Theorem again.[1]

21.2 An aside: 'Gödel sentences' again

Theorem 20.3 tells us what one kind of fixed point for $\neg\mathsf{Prov}_T$ looks like, for it shows that a canonical Gödel sentence fits the bill. But in a number of modern presentations, this is not spelt out. Rather the preferred proof for the First Theorem goes as just sketched, i.e. via the general Diagonalization Lemma, which tells us that there *are* fixed points for $\neg\mathsf{Prov}_T$, combined with the general principles (C1) and (Cω).

Which is technically just fine, of course. But if you do go via this route, you do need to be a bit careful about your *commentary*. In particular, you will get into trouble if (i) you call *any* fixed point for $\neg\mathsf{Prov}_T$ which is therefore undecidable by T a 'Gödel sentence', but (ii) you also claim that the Gödel sentences for T are true-if-and-only-if-unprovable-in-T, and hence (since unprovable) are true. For look again at the second corollary of Theorem 20.4. That shows that if T is an unsound theory, there will be some *false* fixed points for $\neg\mathsf{Prov}_T$. So if T is unsound it will have *false* Gödel sentences in the wide sense of (i).[2]

We've already touched on this theme in Section 16.8. It is canonical Gödel sentences – built up in something like Gödel's original way from a wff that perspicuously recapitulates a p.r. definition of the relation Gdl_T – that indirectly say of themselves that they are unprovable and hence must be true if unprovable (whether T is sound or not). But, once we move away from the canonical cases and start using the idea of a Gödel sentence more generously, then we can't assume that such undecidable sentences always have to be true.[3]

[1] And, for the record, we can of course improve the result in the spirit of Section 19.4, e.g. by using the fact that (Cω) can be weakened to assume only that T is Σ_1 sound. Cf. Section 20.1, fn. 3. We won't pause to spell this out.

[2] So, T entails a wff that says that such a fixed point is true iff unprovable, but – being unsound – it entails a *false* biconditional: the rogue fixed points are unprovable but false!

[3] The need for care on this point has been pressed by Peter Milne: I'm grateful to him for

21.3 The Gödel-Rosser Theorem again

(a) Back to the technicalities! Our next major task is to fulfil our promise to prove the Gödel-Rosser Theorem: we'll show that nice theories are incomplete *without* using the assumption of ω-consistency (or weaker versions of it like 1-consistency).

As we noted in Section 19.3, Rosser's basic trick is to construct a sentence which 'says' *If I'm provable, there's already a proof of my negation.* Here's one way of developing that idea.

Consider the relation $\overline{Prf}_T(m, n)$ which holds when m numbers a T-proof of the *negation* of the wff with number n. This relation is obviously p.r. given that Prf_T is; so assuming T is nice it will be captured by a wff $\overline{\mathsf{Prf}}_T(\mathsf{x}, \mathsf{y})$.[4] So now consider *the Rosser provability predicate* defined as follows:

$$\mathsf{RProv}_T(\mathsf{x}) =_{\text{def}} \exists \mathsf{v}(\mathsf{Prf}_T(\mathsf{v}, \mathsf{x}) \wedge (\forall \mathsf{w} \leq \mathsf{v}) \neg \overline{\mathsf{Prf}}_T(\mathsf{w}, \mathsf{x})).$$

Then a sentence is Rosser-provable in T – its g.n. satisfies the Rosser provability predicate – if it has a proof (in the ordinary sense) and there's no 'smaller' proof of its negation.

(b) So now we apply the Diagonalization Lemma, not to the negation of a regular provability predicate (which is what we just did to get Gödel's First Theorem again), but to the negation of a Rosser provability predicate. The Lemma then tells us that there's a sentence R_T which is a fixed point for $\neg\mathsf{RProv}_T(\mathsf{x})$. That is to say, assuming T is nice,

$$T \vdash \mathsf{R}_T \leftrightarrow \neg\mathsf{RProv}_T(\ulcorner \mathsf{R}_T \urcorner).$$

So if T is sound and its theorems are true, then R_T will indeed be true just so long as it isn't Rosser-provable. In other words, R_T is true just if, if it's provable, there is already a proof of its negation. (So by the argument of Section 19.3, if T is sound, $T \nvdash \mathsf{R}_T$ and $T \nvdash \neg\mathsf{R}_T$.)

(c) We will next prove the following general result, which is the analogue of Theorem 21.1:

> **Theorem 21.2** *Let T be a nice theory, and let γ be* any *fixed point for $\neg\mathsf{RProv}_T(\mathsf{x})$. Then $T \nvdash \gamma$ and $T \nvdash \neg\gamma$.*

showing me pre-publication version of Milne (2007), in which he gives chapter and verse on the sins of various textbooks!

[4] Consider the p.r. function defined by $neg(x) =_{\text{def}} \ulcorner\neg\urcorner \star x$, where '$\star$' is the concatenation function from Section 15.6. We have $neg(\ulcorner\varphi\urcorner) = \ulcorner\neg\urcorner \star \ulcorner\varphi\urcorner = \ulcorner\neg\varphi\urcorner$. So *neg* takes the g.n. of a wff and returns the g.n. of its negation.

Now let's suppose we can introduce a function-symbol into T's language to capture this function (see the end of Section 12.2). We'll use the symbol '$\dot{\neg}$' to do the job. (So now '$\neg$' has a double use: 'undotted' and attached to a wff, it is a truth-functional operator; 'dotted' and attached to a term, it is expresses a corresponding numerical function. The dotting convention will stop us getting confused.) With this neat new notation, we can put $\overline{\mathsf{Prf}}_T(\mathsf{x}, \mathsf{y}) =_{\text{def}} \mathsf{Prf}_T(\mathsf{x}, \dot{\neg}\mathsf{y})$.

The proof is inevitably a bit messy, so feel free to skip: but, if you want the details, here goes:

Proof for first half Suppose γ is a provable fixed point for RProv_T. Then – again dropping subscripts for readability – because γ is provable, for some m, $Prf(m, \ulcorner\gamma\urcorner)$. Since Prf captures Prf, $T \vdash \mathsf{Prf}(\mathsf{m}, \ulcorner\gamma\urcorner)$.

Also, by T's consistency, $\neg\gamma$ is unprovable, so for all n, not-$\overline{Prf}(n, \gamma)$. Since $\overline{\mathsf{Prf}}$ captures $\overline{Prf}$, then for all $n \leq m$ in particular, $T \vdash \neg\overline{\mathsf{Prf}}(\mathsf{n}, \ulcorner\gamma\urcorner)$. Using the result (O4) of Section 9.9, that shows $T \vdash (\forall \mathsf{w} \leq \mathsf{m})\neg\overline{\mathsf{Prf}}(\mathsf{w}, \ulcorner\gamma\urcorner)$.

Putting these results together, $T \vdash \mathsf{Prf}(\mathsf{m}, \ulcorner\gamma\urcorner) \wedge (\forall \mathsf{w} \leq \mathsf{m})\neg\overline{\mathsf{Prf}}(\mathsf{w}, \ulcorner\gamma\urcorner)$. So $T \vdash \mathsf{RProv}_T(\ulcorner\gamma\urcorner)$. But by hypothesis, $T \vdash \gamma \leftrightarrow \neg\mathsf{RProv}_T(\ulcorner\gamma\urcorner)$. Therefore if γ is provable, we also have $T \vdash \neg\mathsf{RProv}_T(\ulcorner\gamma\urcorner)$. Contradiction. So $T \nvdash \gamma$. ⊠

Proof for second half Now suppose $\neg\gamma$ is provable. For some m, $Prf(m, \ulcorner\neg\gamma\urcorner)$, so $T \vdash \overline{\mathsf{Prf}}(\mathsf{m}, \ulcorner\gamma\urcorner)$.

But, by hypothesis, we still have $T \vdash \gamma \leftrightarrow \neg\mathsf{RProv}_T(\ulcorner\gamma\urcorner)$, so since $\neg\gamma$ is provable, it follows that $T \vdash \mathsf{RProv}_T(\ulcorner\gamma\urcorner)$. Hence, unpacking that, $T \vdash \exists \mathsf{x}\,\mathsf{Prf}(\mathsf{x}, \ulcorner\gamma\urcorner) \wedge (\forall \mathsf{w} \leq \mathsf{x})\neg\overline{\mathsf{Prf}}(\mathsf{w}, \ulcorner\gamma\urcorner)$.

Arguing inside T there must be a witness a to this existential quantification, so $\mathsf{Prf}(\mathsf{a}, \ulcorner\gamma\urcorner) \wedge (\forall \mathsf{w} \leq \mathsf{a})\neg\overline{\mathsf{Prf}}(\mathsf{w}, \ulcorner\gamma\urcorner)$. But T proves that $\mathsf{a} \leq \mathsf{m} \vee \mathsf{m} \leq \mathsf{a}$ by (O8) of Section 9.9. So now consider cases.

Suppose $\mathsf{a} \leq \mathsf{m}$. Then since $\mathsf{Prf}(\mathsf{a}, \ulcorner\gamma\urcorner)$, we'll get $\mathsf{Prf}(0, \ulcorner\gamma\urcorner) \vee \mathsf{Prf}(1, \ulcorner\gamma\urcorner) \vee \ldots \vee \mathsf{Prf}(\mathsf{m}, \ulcorner\gamma\urcorner)$, using (O3) of Section 9.9. But by consistency $T \nvdash \gamma$, so for all n, not-$Prf(n, \ulcorner\gamma\urcorner)$, so for all n, $T \vdash \neg\mathsf{Prf}(\mathsf{n}, \ulcorner\gamma\urcorner)$. Contradiction.

Suppose $\mathsf{m} \leq \mathsf{a}$. Then since $(\forall \mathsf{w} \leq \mathsf{a})\neg\overline{\mathsf{Prf}}(\mathsf{w}, \ulcorner\gamma\urcorner)$, it follows that $\neg\overline{\mathsf{Prf}}(\mathsf{m}, \ulcorner\gamma\urcorner)$, and again contradiction.

So the initial supposition that $\neg\gamma$ is provable leads to contradiction either way, and $T \nvdash \neg\gamma$. ⊠

(d) As we said, proving Theorem 21.2 is – if not exactly harder – a lot messier than proving Theorem 21.1. However, we now know that any fixed point for $\neg\mathsf{RProv}_T$ must be formally undecidable in T. But the Diagonalization Lemma has already told us that there has to *be* such a fixed point, R_T. So R_T is formally undecidable (assuming no more than T's niceness).

To complete our proof of the Gödel-Rosser Theorem, we just need to confirm that R_T can be Π_1. By the first corollary to Theorem 20.4, that will be the case if $\neg\mathsf{RProv}_T$ is Π_1, i.e. RProv_T is Σ_1. But it is:

Proof sketch Given the way that Prf and hence $\overline{\mathsf{Prf}}$ are built up by tracking the definition of Prf, it will be quite easy to prove that the Σ_1 wff $\overline{\mathsf{Prf}}(\mathsf{w}, \mathsf{x})$ is equivalent to the Π_1 wff $\forall \mathsf{z}(\overline{\mathsf{Prf}}(\mathsf{w}, \mathsf{z}) \rightarrow \mathsf{z} = \mathsf{x})$. And then – using the quantifier shift trick of Section 13.6 – it follows that $(\forall \mathsf{w} \leq \mathsf{v})\neg\overline{\mathsf{Prf}}_T(\mathsf{w}, \mathsf{x}))$ is Σ_1, and so RProv_T is too. ⊠

(e) So, putting everything together gives us, at last, the promised Gödel-Rosser Theorem:

Theorem 19.6 *If T is a nice theory, then there is an L_A-sentence φ of Goldbach type such that neither $T \vdash \varphi$ nor $T \vdash \neg\varphi$.*

Phew!

21.4 Capturing provability?

Having backtracked to the Diagonalization Lemma, we have re-established the First Theorem and have now proved the stronger Gödel-Rosser Theorem. Now we strike out to get a range of new theorems.

First, consider again the T-wff $\mathsf{Prov}_T(\mathsf{x})$ which *expresses* the property $Prov_T$ of being the g.n. of a T-theorem. The obvious next question to ask is: does this wff also case-by-case *capture* that property?

No, it doesn't, because in fact

Theorem 21.3 *No open wff in a nice theory T can capture the corresponding numerical property* Prov_T.

Proof Suppose for reductio that $\mathsf{P}(\mathsf{x})$ abbreviates an open wff – not necessarily identical to $\mathsf{Prov}_T(\mathsf{x})$ – which captures $Prov_T$. By the Diagonalization Lemma applied to $\neg\mathsf{P}(\mathsf{z})$, there is some wff γ such that

1. $T \vdash \gamma \leftrightarrow \neg\mathsf{P}(\ulcorner\gamma\urcorner)$.

By the general assumption that P captures $Prov_T$, we have in particular

2. if $T \vdash \gamma$, i.e. $Prov_T(\ulcorner\gamma\urcorner)$, then $T \vdash \mathsf{P}(\ulcorner\gamma\urcorner)$,
3. if $T \nvdash \gamma$, i.e. not-$Prov_T(\ulcorner\gamma\urcorner)$, then $T \vdash \neg\mathsf{P}(\ulcorner\gamma\urcorner)$.

Contradiction quickly follows. By (3) and (1), if $T \nvdash \gamma$, then $T \vdash \gamma$. Hence $T \vdash \gamma$. So by (2) and (1) we have both $T \vdash \mathsf{P}(\ulcorner\gamma\urcorner)$ and $T \vdash \neg\mathsf{P}(\ulcorner\gamma\urcorner)$ making T inconsistent, contrary to hypothesis. ⊠

Hence $Prov_T$ cannot be captured in T: and so – answering our original question – $\mathsf{Prov}(\mathsf{x})_T$ in particular doesn't capture that property.

And here's a quick corollary of our theorem. If T is nice, it includes Q and is p.r. adequate, and so can capture any p.r. property. So it immediately follows that *the provability property $Prov_T$ for any nice theory is not primitive recursive*, which is to say that there is no p.r. function of n which returns 0 if $Prov_T(n)$ is true, and 1 otherwise. (Later, we'll show that $Prov_T$ is not a decidable property at all – see Section 30.4; but of course we can't do that now, as haven't yet got a general theory of what makes for a decidable property.)

Which all establishes two general points. (i) There can be properties like $Prov_T$ which are expressible in a given theory T but not capturable. (ii) And although every p.r. property and relation can be expressed by an open Σ_1 wff, there are open Σ_1 wffs like $\mathsf{Prov}(\mathsf{x})$ which don't express p.r. properties or relations.

21.5 Tarski's Theorem

Next, we visit twin peaks which can also be reached via the Diagonalization Lemma. The path is very straightforward, but it leads to a pair of rather spectacular results that are usually packaged together as *Tarski's Theorem*.[5]

(a) Recall a familiar thought: 'snow is white' is true iff snow *is* white. Likewise for all other sensible replacements for 'snow is white'. In sum, every instance of *'φ' is true iff* φ is true. And that's because of the meaning of the informal truth-predicate 'true'.

Now suppose we have fixed on some scheme for Gödel numbering wffs of the interpreted arithmetical language L. Then we can define a corresponding numerical property *True* as follows:

> *True*(n) is true iff n is the g.n. of a true sentence of L.

Suppose that the open wff $\mathsf{T}(\mathsf{x})$ – which belongs to an arithmetical language L' which includes L – expresses this numerical property *True*. Then, for any L-sentence φ,

> φ is true iff *True*$(\ulcorner\varphi\urcorner)$ iff $\mathsf{T}(\ulcorner\varphi\urcorner)$ is true.

Hence, for any L-sentence φ, every corresponding L'-sentence

$$\mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$$

is true.

So this motivates our first main definition:

i. An open L'-wff $\mathsf{T}(\mathsf{x})$ is a *formal truth-predicate for* L iff for every L-sentence φ, $\mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ is true.

Here's a companion definition:

ii. A theory T (with language L' which includes L) is a *truth-theory for* L iff for some L'-wff $\mathsf{T}(\mathsf{x})$, $T \vdash \mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ for every L-sentence φ.

Equally often, a truth-theory for L is called a 'definition of truth for L'.

(b) Suppose T is a nice arithmetical theory with language L. An obvious question arises: could T be competent to define truth *for its own language* (i.e., can T include a truth-theory for L)? And the answer is immediate:

> **Theorem 21.4** *No nice theory can define truth for its own language.*

[5] Alfred Tarski investigated these matters in his classic (1933); though Gödel again had already noted the key point, e.g. in a letter to Zermelo written in October, 1931 (Gödel, 2003b, pp. 423–429). Also see the quotation later in the next section.

Proof Assume T defines truth for L, i.e. there is an L-predicate $\mathsf{T}(\mathsf{x})$ such that $T \vdash \mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ for every L-sentence φ. Since T is nice, the Diagonalization Lemma applies, so applying the Lemma to the negation of $\mathsf{T}(\mathsf{x})$, we know that there must be some sentence L – a Liar sentence! – such that

1. $T \vdash \mathsf{L} \leftrightarrow \neg\mathsf{T}(\ulcorner\mathsf{L}\urcorner)$.

But, by our initial assumption, we also have

2. $T \vdash \mathsf{T}(\ulcorner\mathsf{L}\urcorner) \leftrightarrow \mathsf{L}$.

It is immediate that T is inconsistent and so not nice, contrary to hypothesis. So our assumption must be wrong: T can't define truth for its own language. ⊠

(c) That first theorem puts limits on what a nice theory can *prove* about truth. But with very modest extra assumptions, we can put limits on what a theory's language can even *express* about truth.

Consider L_A for the moment, and suppose that there is an L_A truth-predicate T_A that expresses the corresponding truth property $True_A$. Since Q is nice, the Diagonalization Lemma applies, in particular to the negation of $\mathsf{T}_A(\mathsf{x})$. So we know that for some L_A sentence L,

1. $\mathsf{Q} \vdash \mathsf{L} \leftrightarrow \neg\mathsf{T}_A(\ulcorner\mathsf{L}\urcorner)$.

But (and here comes the extra assumption we said we were going to invoke!) everything Q proves is true, since Q's axioms are of course true and its logic is truth preserving. So

2. $\mathsf{L} \leftrightarrow \neg\mathsf{T}_A(\ulcorner\mathsf{L}\urcorner)$

will also be a true L_A wff. But, by the assumption that T_A is a truth-predicate for L_A,

3. $\mathsf{T}_A(\ulcorner\mathsf{L}\urcorner) \leftrightarrow \mathsf{L}$

must be true too. (2) and (3) immediately lead to contradiction again. Therefore our supposition that T_A is a truth-predicate has to be rejected. Hence no predicate of L_A can even express the numerical property $True_A$.

The argument evidently generalizes. Take any language L rich enough for us to be able to formulate in L something equivalent to the very elementary arithmetical theory Q (that's so we can prove the Diagonalization Lemma again). Call that an arithmetically adequate language. Then by the same argument, assuming Q is a correct theory,

> **Theorem 21.5** *No predicate of an arithmetically adequate language L can express the numerical property* True_L *(i.e. the property of numbering a truth of L).*

This tells us that while you can express *syntactic* properties of a sufficiently rich formal theory of arithmetic (like provability) inside the theory itself via Gödel

numbering, you can't express some key *semantic* properties (like arithmetical truth) inside the theory.

(d) Suppose T is a nice theory. Then (1) there are some numerical properties that T can capture (the p.r. ones for a start); (2) there are some properties that T can express but not capture (for example $Prov_T$); and (3) there are some properties that T's language L cannot even express (for example $True_L$, the numerical property of numbering-a-true-L-wff).

It is not, we should hasten to add, that the property $True_L$ is mysteriously ineffable, and escapes all formal treatment. A richer theory T' with a richer language L' may perfectly well be able to capture $True_L$. But the point remains that, however rich a given theory of arithmetic is, there will be limitations, not only on what numerical properties it can capture but even on which numerical properties that particular theory's language can express.

21.6 The Master Argument

Our results about the non-expressibility of truth of course point to another, particularly illuminating, take on the argument for incompleteness.

For example: truth in L_A isn't provability in PA, because while PA-provability *is* expressible in L_A, truth-in-L_A *isn't*. So assuming that PA is sound and everything provable in it is true, this means that there must be truths of L_A which it can't prove. Similarly, of course, for other nice theories.

And in a way, we might well take this to be *the* Master Argument for incompleteness, revealing the roots of the phenomenon.[6] Gödel himself wrote (in response to a query)

> I think the theorem of mine that von Neumann refers to is ... that a complete epistemological description of a language A cannot be given in the same language A, because the concept of truth of sentences in A cannot be defined in A. *It is this theorem which is the true reason for the existence of undecidable propositions in the formal systems containing arithmetic.* I did not, however, formulate it explicitly in my paper of 1931 but only in my Princeton lectures of 1934. The same theorem was proved by Tarski in his paper on the concept of truth [Tarski (1933)].[7]

In sum, as we emphasized before, arithmetical truth and provability in this or that formal system must peel apart.

[6] Or if not *the* argument, at least one of the proofs which belongs in The Book: see Section 5.4.

[7] The emphasis is mine. Gödel's letter is quoted in Feferman (1984), which also has a very interesting discussion of why Gödel chose not to highlight this line of argument for incompleteness in his original paper, a theme we'll return to in Chapter 28. The passage in the Princeton lectures to which Gödel refers is at Gödel (1934, p. 363).

21.7 The length of proofs

Finally in this action-packed chapter, I can't resist adding a couple of rather nice related results about the length of proofs. However, these are results which you can certainly skip at a first reading – even though it would be good to return to them at some point, because they are ingenious and instructive.

(a) We can crudely indicate the length of a wff φ by its g.n.; and likewise we can indicate the length of a proof Π by *its* (super) Gödel number. To be sure, those aren't the most natural measures: counting the symbols in φ and the number of symbols or the number of wffs in Π would be better. However our rough-and-ready indicators are particularly simple to work with.[8]

Take a specific theory T. As a general tendency, we'll expect that the longer a wff, the longer its T-proof (if it has one). But is there any tidy order in this relationship? Suppose f is some function: then we'll say that a proof for φ is *f-bounded* if the proof's g.n. is less than $f(\ulcorner\varphi\urcorner)$. Then it would indeed be rather tidy if, for a given theory T, there were some corresponding p.r. function len_T which puts a general ceiling on the length of T-proofs – meaning that, for *any* φ, if it is a provable at all, it has a len_T-bounded proof. However, unfortunately,

> **Theorem 21.6** *If T is nice theory, then for any p.r. function f, there is a provable wff φ which has no f-bounded proof.*

Proof sketch Suppose the theorem is false, i.e. suppose that there is a p.r. ceiling function len_T such that for any φ, if it is T-provable at all, it has a len_T-bounded proof. Then there would be a p.r. procedure for testing whether φ has a proof in T. Just calculate $len_T(\ulcorner\varphi\urcorner)$, and do a bounded search using a 'for' loop to run through all the possible proofs up to that size to see if one of them is in fact a proof of φ. But the easy corollary to Theorem 21.3 tells us that there can be no p.r. procedure for deciding whether φ is a theorem. So there can be no such function len_T. ⊠

But now note that some p.r. functions $f(n)$ grow fantastically fast and soon have huge values even for low values of n. So our theorem means, roughly speaking, that there will inevitably be theorems which can be stated briefly but which only have relatively enormous proofs.

(b) Our result that some short wffs have relatively enormous proofs is not too surprising. But our next result is perhaps more exciting.

Let's say that a theory T_1 *exhibits speed-up over* T_2 iff for *any* p.r. function f, there is some wff φ such that (i) both $T_1 \vdash \varphi$ and $T_2 \vdash \varphi$ but (ii) while there is a T_1-proof of φ with g.n. p, there is no T_2-proof with g.n. less than or equal to

[8] And it is reasonably easy to check that, at the price of some minor complications, the arguments we are going to use in order to establish results about the length of proofs on our somewhat unnatural measuring scheme can be carried over to establish parallel results about the length of proofs as measured in more natural ways.

$f(p)$. In other words, there are indefinitely many wffs for which T_1 gives 'much shorter' proofs than T_2. Then we have the following:

> **Theorem 21.7** *If T is nice theory, and γ is some sentence such that neither $T \vdash \gamma$ nor $T \vdash \neg\gamma$, then the theory $T + \gamma$ got by adding γ as a new axiom exhibits speed-up over T.*

This means that adding a previously undecidable wff γ to a nice theory T as a new axiom not only enables us to prove *new* theorems (γ, for a start) but it also radically shortens the proofs of *old* theorems. The general phenomenon was first noted in an abstract by Gödel (1936).[9]

Number theorists have long been familiar with cases where arithmetical theorems seem only to have very long and messy proofs in 'pure' arithmetic even though there are relatively nice proofs if we are allowed to extend arithmetic by e.g. the theory of complex analysis. What our theorem shows is that there is an inevitability about this kind of situation.

Proof sketch Suppose the theorem is false. So there is a sentence γ which is undecided by T, and there is also a p.r. function f such that for every wff φ, if φ has a proof in $T + \gamma$ with g.n. p, then it has a proof in the original T with g.n. number no greater than $f(p)$.

Now, for any wff φ, $(\gamma \vee \varphi)$ is trivially provable in $T + \gamma$. And there will be a very simple computation, with no open-ended searching, that takes us from the g.n. of φ to the g.n. of the trivial proof of $(\gamma \vee \varphi)$. In other words, the g.n. of the proof will be $h(\ulcorner\varphi\urcorner)$, for some p.r. function h.[10] So, by our supposition, $(\gamma \vee \varphi)$ must have a proof in T with g.n. no greater than $f(h(\ulcorner\varphi\urcorner))$.

Next consider the theory $T + \neg\gamma$. Trivially again, for any φ, $T + \neg\gamma \vdash \varphi$ iff $T \vdash (\gamma \vee \varphi)$. So we have a p.r. decision procedure for telling whether an arbitrary φ is a theorem of $T + \neg\gamma$. Just run a 'for' loop examining in turn all the T-proofs with g.n. up to $f(h(\ulcorner\varphi\urcorner))$ and see if a proof of $(\gamma \vee \varphi)$ turns up.

But $T + \neg\gamma$ is still a nice theory: it is consistent (else we'd have $T \vdash \gamma$, contrary to hypothesis), it is p.r. axiomatized, and it contains Q since T does. So there can't be a p.r. procedure for testing theoremhood in $T + \neg\gamma$. Contradiction. ⊠

(c) The informal arguments above for our two theorems in this section are fine as far as they go. However, there is some interest in noting that we can use the Diagonalization Lemma to prove the first of them 'constructively'. That is to say, we can actually give a recipe which takes an arbitrary p.r. function f, and constructs a wff φ which is provable but has no f-bounded proof. (To be quite

[9] See also Section 22.8.

[10] To take the simplest case, imagine $T + \gamma$'s logical system is set up with the rule of $\vee$-introduction, so the little sequence γ, $(\gamma \vee \varphi)$ will serve as the needed proof. Then the super g.n. of this proof is

$$2^{\ulcorner\gamma\urcorner} \cdot 3^{\ulcorner(\gamma \vee \varphi)\urcorner} = 2^{\ulcorner\gamma\urcorner} \cdot 3^{\ulcorner(\urcorner \star \ulcorner\gamma\urcorner \star \ulcorner\vee\urcorner \star \ulcorner\phi\urcorner \star \ulcorner)\urcorner}.$$

Evidently the proof's g.n. is then a p.r. function of $\ulcorner\phi\urcorner$.

honest, these details are more than you can really need to know: but they're just too pretty to miss out!)

Suppose then that the *Prf* relation for T is captured by the open wff $\mathsf{Prf}(\mathsf{x}, \mathsf{y})$, and that our arbitrary p.r. function f is captured by $\mathsf{F}(\mathsf{x}, \mathsf{y})$. Now form the wff

$$\forall \mathsf{v}\{\mathsf{F}(\mathsf{z}, \mathsf{v}) \to (\forall \mathsf{u} \leq \mathsf{v})\neg \mathsf{Prf}(\mathsf{u}, \mathsf{z})\}.$$

By the Diagonalization Lemma, there is a wff φ such that

$$T \vdash \varphi \leftrightarrow \forall \mathsf{v}\{\mathsf{F}(\ulcorner \varphi \urcorner, \mathsf{v}) \to (\forall \mathsf{u} \leq \mathsf{v})\neg \mathsf{Prf}(\mathsf{u}, \ulcorner \varphi \urcorner)\}.$$

We'll show that (i) $T \vdash \varphi$, but also (ii) the g.n. of the 'smallest' proof of φ is greater than $f(\ulcorner \varphi \urcorner)$.

Suppose, for reductio, that there *is* a proof of φ with g.n. $p \leq f(\ulcorner \varphi \urcorner) = k$. Then we are assuming

1. $T \vdash \varphi$;
2. for some $p \leq k$, $Prf(p, \ulcorner \varphi \urcorner)$, so $T \vdash \mathsf{Prf}(\mathsf{p}, \ulcorner \varphi \urcorner)$.

By (1), $T \vdash \forall \mathsf{v}\{\mathsf{F}(\ulcorner \varphi \urcorner, \mathsf{v}) \to (\forall \mathsf{u} \leq \mathsf{v})\neg \mathsf{Prf}(\mathsf{u}, \ulcorner \varphi \urcorner)\}$. So instantiating the quantifier we get $T \vdash \mathsf{F}(\ulcorner \varphi \urcorner, \mathsf{k}) \to (\forall \mathsf{u} \leq \mathsf{k})\neg \mathsf{Prf}(\mathsf{u}, \ulcorner \varphi \urcorner)$.

But since $f(\ulcorner \varphi \urcorner) = k$ and we're assuming F captures f, $T \vdash \mathsf{F}(\ulcorner \varphi \urcorner, \mathsf{k})$. Hence $T \vdash (\forall \mathsf{u} \leq \mathsf{k})\neg \mathsf{Prf}(\mathsf{u}, \ulcorner \varphi \urcorner)$. And now we can manipulate Result 3 about Q (and hence T) proved in Section 9.9 to deduce that for any $p \leq k$, $T \vdash \neg \mathsf{Prf}(\mathsf{p}, \ulcorner \varphi \urcorner)$. Which contradicts (2).

That establishes (ii), i.e. there is no proof of φ with g.n. $p \leq f(\ulcorner \varphi \urcorner) = k$. So, for each $p \leq k$, $Prf(p, \ulcorner \varphi \urcorner)$ is false, hence $T \vdash \neg \mathsf{Prf}(\mathsf{p}, \ulcorner \varphi \urcorner)$. So by Result 5 from Section 9.9, $T \vdash (\forall \mathsf{u} \leq \mathsf{k})\neg \mathsf{Prf}(\mathsf{u}, \ulcorner \varphi \urcorner)$. But since F captures f, $T \vdash \forall \mathsf{v}(\mathsf{F}(\ulcorner \varphi \urcorner, \mathsf{v}) \leftrightarrow \mathsf{v} = \mathsf{k})$, whence $T \vdash \forall \mathsf{v}\{\mathsf{F}(\ulcorner \varphi \urcorner, \mathsf{v}) \to (\forall \mathsf{u} \leq \mathsf{v})\neg \mathsf{Prf}(\mathsf{u}, \ulcorner \varphi \urcorner)\}$ and thus finally (i) $T \vdash \varphi$. ⊠

22 Second-order arithmetics

As we noted in Section 10.1, the intuitive principle of mathematical induction looks to be a *second-order* principle that quantifies over numerical properties, and which can't be directly expressed in a first-order theory that only quantifies over numbers. So you might well be wondering: why not work with a second-order arithmetic, rather than hobble ourselves by artificially forcing our formal arithmetic into a first-order straightjacket? True, we now know that – so long as it stays consistent and properly axiomatized – a richer theory won't entirely escape the reach of the Gödel-Rosser Theorem, any more than a first-order theory can. But still, we ought to say at least a little about second-order arithmetics.

Indeed, there is a pressing issue about such theories which really needs to be addressed head on at this point. For if you have done a standard university algebra course, you might very well be feeling pretty puzzled by now. Such a course typically introduces axioms for some version of 'Second-order Peano Arithmetic', and there is an elementary textbook proof that these axioms pin down a unique type of structure. But if this second-order arithmetic *does* pin down the structure of the natural numbers, then – given that any arithmetic sentence makes a determinate claim about this structure – it apparently follows that this theory does enough to settle the truth-value of every arithmetic sentence. Which makes it sound as if there *can* after all be a (consistent) negation-complete axiomatic theory of arithmetic richer than first-order PA, flatly contradicting the Gödel-Rosser Theorem.

So just what is going on here? In this chapter, we say enough to sort things out (though, in order not to take us too far from the main concerns of this book, a number of relevant results will just be stated rather than proved).

22.1 Second-order arithmetical languages

As a preliminary, we need to characterize some suitable language(s) for second-order arithmetics.

Start from our familiar first-order language L_A, i.e. $\langle \mathcal{L}_A, \mathcal{I}_A \rangle$. We first extend the syntax $\mathcal{L}_A$ by adding second-order quantifiers to get $\mathcal{L}_{2A}$. We then augment the semantics $\mathcal{I}_A$ to deal with the new quantifiers: doing this in the most obvious and natural way gives us the so-called 'full' second-order interpretation $\mathcal{I}_{2A}$. So we'll define L_{2A} to be the resulting language $\langle \mathcal{L}_{2A}, \mathcal{I}_{2A} \rangle$. But as we'll also note, there are other ways of giving a semantics to $\mathcal{L}_{2A}$: we'll briefly mention just one.

Now for some details.

(a) Syntax first. The key new idea is as follows. In $\mathcal{L}_A$, the first-order variables $\mathsf{x}, \mathsf{y}, \mathsf{z}, \ldots$ can appear in the place of names; in $\mathcal{L}_{2A}$ we also have second-order variables $\mathsf{X}, \mathsf{Y}, \mathsf{Z}, \ldots$, where these can replace predicates. We'll concentrate here on the case of second-order variables that act like monadic, i.e. *one*-place, predicates.[1] Both sorts of variable will have associated quantifiers.

In $\mathcal{L}_A$, there is just one type of atomic wff, of the form $\sigma = \tau$, where σ and τ are terms. But now $\mathcal{L}_{2A}$ also has a second type of atomic wff. The new rule is: if Ξ stands in for some second-order variable and τ is a term, then $\Xi\tau$ is an atomic wff. For example,

$$\mathsf{X0}, \mathsf{Xx}, \mathsf{XSSSx}, \mathsf{Y}(\mathsf{S0} + \mathsf{z}), \mathsf{Z}((\mathsf{x} + \mathsf{y}) \times \mathsf{SS0})$$

are all atomic wffs of the new type.

More complex wffs of $\mathcal{L}_A$ are built up from atomic wffs by using connectives and/or prefixing first-order quantifiers like $\forall\mathsf{x}$, $\exists\mathsf{y}$. And the sentences of $\mathcal{L}_A$ are the closed wffs, i.e. wffs without unquantified variables dangling free. Similarly, more complex wffs of $\mathcal{L}_{2A}$ are built up from atomic wffs by using connectives and/or prefixing first-order quantifiers and/or prefixing second-order quantifiers like $\forall\mathsf{X}$, $\exists\mathsf{Y}$ (note then that every old $\mathcal{L}_A$ wff is also an $\mathcal{L}_{2A}$ wff). And the sentences of $\mathcal{L}_{2A}$ are the closed wffs, i.e. wffs without unquantified variables of either type dangling free. For example, the following are sentences of $\mathcal{L}_{2A}$:

$$\exists\mathsf{X}\,\mathsf{X0},\ \forall\mathsf{X}(\forall\mathsf{x}\mathsf{Xx} \rightarrow \mathsf{X0}),\ \exists\mathsf{X}(\neg\mathsf{XS0} \wedge \forall\mathsf{x}(\mathsf{Xx} \rightarrow \mathsf{XSSx})).$$

(b) Now for the semantics, again just giving the headline news and concentrating on the quantifiers. When ξ is a first-order variable, an $\mathcal{L}_{2A}$-sentence of the form $\forall\xi\varphi(\xi)$ says that every number satisfies the condition expressed by $\varphi(\xi)$; and a sentence of the form $\exists\xi\varphi(\xi)$ says that at least one number satisfies the condition expressed by $\varphi(\xi)$. Similarly, when Ξ is a second-order variable, an $\mathcal{L}_{2A}$-sentence of the form $\forall\Xi\psi(\Xi)$ says that every numerical property satisfies the condition expressed by $\psi(\Xi)$; and a sentence of the form $\exists\Xi\psi(\Xi)$ says that at least one numerical property satisfies the condition expressed by $\psi(\Xi)$.

For example: $\exists\mathsf{X}\,\mathsf{X0}$ says that at least one property is had by the number zero. $\forall\mathsf{X}(\forall\mathsf{x}\mathsf{Xx} \rightarrow \mathsf{X0})$ says that, for any property, if every number has it, then zero in particular has it – another trivial truth. And $\exists\mathsf{X}(\neg\mathsf{XS0} \wedge \forall\mathsf{x}(\mathsf{Xx} \rightarrow \mathsf{XSSx}))$ says that there is a property lacked by 1 such that, if any number n has it, so does $n + 2$. That's a third truth: the property of *being even* is an obvious instance.

[1] For more details see e.g. Shapiro (1991, Ch. 3). If we were doing all this properly, we'd follow Shapiro by also including at the outset dyadic, triadic, etc. variables than can stand in for two-place, three-place, etc. relational predicates, with all *their* associated quantifiers too. Next we'd prove that even a very modest arithmetical theory can handle numerical codes for ordered pairs, triples etc. of numbers (cf. Section 2.5). We can then replace two-place predicates with one-place predicates that apply to codes-for-pairs, and n-place predicates with one-place predicates that apply to codes-for-n-tuples. So, in fact, we can then show that an arithmetic with the full apparatus of quantifiers running over n-place relations for any n is equivalent to one with just monadic second-order quantifiers. Which is why it is legitimate to cut corners here and concentrate on the monadic case, ignoring the rest.

(c) Note, however, that our initial semantic story doesn't yet fully pin down the interpretation of $\mathcal{L}_{2A}$. For just which numerical properties *are* we quantifying over here? As before, we are thinking of properties extensionally (see Section 4.2). So quantifying over properties is in effect quantifying over their extensions, where property-extensions are subsets of the domain of objects – i.e., in the present case, subsets of the domain $\mathbb{N}$. Our question therefore comes to this: which extensions should we recognize as being available for us to quantify over when we use second-order numerical quantifiers? In other words, which subsets of $\mathbb{N}$ do we want to be in the range of the second-order variables?[2]

Well, now the question has been raised, the obvious answer is: *every arbitrary finite or infinite subset of the numbers*. And that's the answer we'll take to be built into the interpretation $\mathcal{I}_{2A}$.

Yet we might well hesitate over the idea of arbitrary infinite sets of numbers – sets which are supposedly perfectly determinate but are in the general case beyond any possibility of our specifying their members. We can readily make sense of the membership of a set of numbers being determined by possession of some characterizing property which gives a recipe for picking out the numbers; and we can readily make sense of the membership being merely stipulated (more or less arbitrarily). However, the first idea gives us infinite sets but not arbitrary ones; and the second idea may give us arbitrary sets (whose members share nothing but the gerrymandered property of having being selected for membership) but not infinite ones – unless we are prepared to conceive of a completed infinite series of arbitrary choices. Neither initial way of thinking of sets uncontentiously makes sense of the classical idea of arbitrary infinite sets of numbers.

Now, in pointing this out, I'm not arguing that we should be sceptical about the classical idea we've just built into $\mathcal{I}_{2A}$. Rather, for present purposes, the moral is simply this: *if we do interpret the second-order quantifiers as ranging over all arbitrary subsets of the domain of numbers, then this commits us to making sense of a clearly infinitary conception*, one that goes beyond anything that is given just in our ordinary understanding of elementary arithmetic.

But let's suppose we are happy with this. Then, in sum, the corresponding 'full' interpretation $\mathcal{I}_{2A}$ for our second-order arithmetical language $\mathcal{L}_{2A}$ starts from $\mathcal{I}_A$ again – i.e. the domain of (first-order) quantification remains $\mathbb{N}$, and '0', 'S', '+' and '×' get the same interpretation as before. And now we add the crucial stipulation that the second-order quantifiers run over the full collection of all arbitrary subsets of the domain.

(d) What about *semantic entailments* between sentences of L_{2A}?

Recall the general definition of semantic entailment. For sentences in the language L, $\Sigma \vDash \varphi$ (the sentences in Σ entail φ) iff every admissible (re)interpretation which makes all the sentences in Σ true makes φ true too. Here, the admissible

[2] As we noted before, in Section 10.8, fn. 11, the term 'set' is now rather firmly linked to a particular industrial-strength iterative conception of sets. So let's stress that we are not committed to thinking of property-extensions as sets in any such heavy-duty sense.

interpretations are those which keep fixed the meanings of the logical apparatus of L, while varying the non-logical aspects of the interpretation.

So, in particular, $\Sigma \models_2 \varphi$ (the L_{2A} sentences in Σ entail φ) iff every admissible interpretation which makes all the sentences in Σ true makes φ true too. As we move from $\mathcal{I}_{2A}$ to other admissible (re)interpretations of L_{2A}, we vary the domain of quantification and/or the interpretations of the non-logical symbols but keep fixed the interpretations of the connectives, first-order quantifiers and the identity sign. We *also* keep fixed $\mathcal{I}_{2A}$'s interpretation of the second-order quantifiers as running over the full collection of *all* arbitrary subsets of the domain, whatever the domain now is.

(e) If the interpretation $\mathcal{I}_{2A}$ makes play with the infinitary idea of arbitrary subsets of the numbers, is there some alternative, less infinitary, way of understanding the second-order quantifiers? Well, if we want to stick closer to the conceptual resources of ordinary elementary arithmetic, one way to go would be to adopt what we'll label $\mathcal{I}_{2a}$, which interprets the second-order quantifiers as ranging only over those non-arbitrary sets of numbers that can be characterized in purely arithmetical terms (more precisely, as ranging over the *arithmetical sets* which can be given as the extensions of purely arithmetical predicates which lack second-order quantifiers). We can't say much more about this sort of interpretation here: but we mention it to highlight the very important point that there *are* choices to be made in how we understand the second-order quantifiers.

22.2 The Induction Axiom

Once we are using a language with second-order quantifiers, we can render the intuitive induction principle as a single sentence which looks like this:

Induction Axiom $\forall \mathsf{X}(\{\mathsf{X0} \wedge \forall \mathsf{x}(\mathsf{Xx} \rightarrow \mathsf{XSx})\} \rightarrow \forall \mathsf{x}\mathsf{Xx})$

But – to take up again the point we've just highlighted – note that a sentence like this does *not* wear its meaning on its face.

There's a famous, often quoted, remark due to Georg Kreisel:

> A moment's reflection shows that the evidence of the first-order schema derives from the second-order [axiom]. (Kreisel, 1972, p. 148)

And this much may be right: we are prepared to give a blanket endorsement to all the instances of the first-order Schema because we accept the intuitive thought that if zero has some property, and if that property is passed down from one number to the next, then all numbers have that property, whatever property that is (compare our introduction of the Induction Schema in Section 10.1). However, there is no obvious reason to suppose that our intuitive thought here aims to generalize over more than some 'natural' class of arithmetical properties (e.g. those with arithmetical sets as extensions), or that it already involves the extended conception of properties whose extensions are quite arbitrary subsets

of the numbers.[3] In other words, the intuitive thought arguably falls well short of the content of the formal Induction Axiom *when interpreted as a sentence of* L_{2A}, i.e. when interpreted as quantifying over the full collection of arbitrary sets of numbers. We shouldn't slide without comment from the intuitive induction principle to the formal Induction Axiom interpreted in that strong way.

22.3 Neat arithmetics

(a) Let's put questions of semantics on hold for the moment, and consider next what happens if we add the second-order Induction Axiom to Q to give us an axiomatized second-order arithmetic whose syntax is $\mathcal{L}_{2A}$. But of course, we won't get far with this axiomatized theory unless we now add some formal proof apparatus to handle our new second-order quantifiers! In particular, we'll need to be able to move from the general Induction Axiom

$$\forall \mathsf{X}(\{\mathsf{X0} \wedge \forall \mathsf{x}(\mathsf{Xx} \to \mathsf{XSx})\} \to \forall \mathsf{x}\mathsf{Xx})$$

to various particular truths of the form

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

How are we going to effect this?

Informally, we can think of this move as involving two steps. (1) We assume that the wff $\varphi(\mathsf{x})$ expresses some property. Then (2) we use the Induction Axiom which applies to *all* properties, and derive the instance for *this* property.

Formally, step (1) amounts to accepting an instance of the so-called

Comprehension Schema $\exists \mathsf{X}\forall \mathsf{x}(\mathsf{Xx} \leftrightarrow \varphi(\mathsf{x}))$

which says that there is a property which is had by just the things which satisfy the condition φ.

Now, in the first-order case, the (basic or derived) 'elimination' rule for extracting information from an existential quantification has the form

from $\exists \xi \psi(\xi)$ and $\psi(\zeta) \to \chi$, infer χ,

where χ doesn't involve the first-order variable/parameter ζ.[4] The rule for extracting information from a second-order existential quantification like an instance of Comprehension is exactly parallel:

from $\exists \Xi \psi(\Xi)$ and $\psi(Z) \to \chi$, infer χ

[3] More careful mathematicians are alert to this point. For example, in his classic survey of modern mathematics, Saunders Mac Lane quite explicitly first presents the second-order principle of induction as an induction over the natural arithmetical properties that can be identified by first-order formulae. And he then distinguishes this intuitive 'induction over properties' (as he calls it) from the stronger 'induction over [arbitrary] sets'. See Mac Lane (1986, p. 44).

[4] Let's not fuss too much about restrictions needed to avoid clash of variables!

where χ doesn't involve the second-order variable/parameter Z. Hence, in order to make use of instances of Comprehension, we need to be able to reason with wffs which have free second-order variables. In particular – and this is what step (2) amounts to – we need to be able to instantiate the Induction Axiom with a free variable Z. So the added rule we want is the universal 'elimination' rule:

> given $\forall \Xi \psi(\Xi)$ (e.g. the Induction Axiom), we can infer $\psi(Z)$

where Z is a second-order variable.

(b) Given those brisk remarks by way of motivation, we can now helpfully present deductive systems for second-order arithmetic as having two parts.

To start with, we set down core logical principles which stay in place across various possible alternative systems. These will include the familiar principles for dealing with connectives and the first-order quantifiers. And we'll add some parallel rules for the second-order quantifiers too – including the two 'elimination' rules we've just mentioned.

Then, crucially, we add a separate comprehension principle: its exact formulation will reflect our view about which expressions express properties/extensions that really are in the intended range of the second-order variables. But, putting it generally, first we fix on a class C of open $\mathcal{L}_{2A}$ wffs. Then we can say that (the universal closures of) instances of the Comprehension Schema are to be axioms, when $\varphi(\mathsf{x})$ belongs to the class C.[5] That tells us that there is a genuine property which applies to just those things which satisfy any given condition φ in C: and as we vary the class C of wffs that are permitted to appear in the comprehension scheme, we get different levels of commitment to properties/extensions.

Putting the ingredients together, here's the key definition we want:

> A *neat* formal system of second-order arithmetic is an axiomatized theory which satisfies the following conditions. (1) Its syntax is $\mathcal{L}_{2A}$. (2) Its basic general logic is a standard deductive system with the usual first-order logical axioms and rules plus their second-order counterparts (but restricted so that we can only instantiate universal second-order quantifications with variables). And (3) the special additional axioms are the axioms of Q, plus the Induction Axiom, plus (the universal closures of) instances of the Comprehension Schema for wffs $\varphi(\mathsf{x})$ belonging to some class C – where it is decidable which wffs belong to C (in order to ensure that a neat theory remains properly axiomatized).

22.4 Introducing PA_2

We'll now introduce one version of Second-Order Peano Arithmetic, PA_2, which is the *strongest* neat theory in the language L_{2A}.

[5] And, to avoid clash of variables, where $\varphi(\mathsf{x})$ does not already contain 'X' free.

So its syntax is $\mathcal{L}_{2A}$; and the theory's built-in interpretation is $\mathcal{I}_{2A}$ – its quantifiers are to be read as running over the full collection of subsets of $\mathbb{N}$. It has the logical rules shared by any neat arithmetic. And its comprehension principle is maximally generous; it allows us to substitute *any* wff $\varphi(\mathsf{x})$ of $\mathcal{L}_{2A}$ into the Comprehension Schema.[6]

Now, you might wonder whether we can really make sense of the formal theory PA_2. For its comprehension principle allows us to define a numerical property by reference to a wff $\varphi(\mathsf{x})$ *which itself might involve second-order quantifiers*: and those quantifiers are supposed to range over the totality of numerical properties, including the one we are defining by the comprehension principle! Can this be legitimate? How can we define something in terms of a collection which is supposed already to include the very item we are trying to define?

Russell famously thought that this involves a vicious circle.[7] The obvious response, however, is Gödel's, which echoes an earlier discussion by Frank Ramsey. This sort of apparently circular definition is in fact unproblematic if we assume that 'the totality of all properties [of numbers] exists somehow independently of our knowledge and our definitions, and that our definitions merely serve to pick out certain of these previously existing properties'.[8] Putting it in terms of property-extensions, if we assume that there already exists a fixed totality of arbitrary infinite subsets of the domain, there is no problem about the idea that we might locate a particular member of that totality by reference to a quantification over it – any more than there is a problem about the idea that we might locate a particular man as the tallest of all the men in the room (including him!).

Now, the official semantic interpretation $\mathcal{I}_{2A}$ that we have built into PA_2's language in effect embodies this Ramsey/Gödel line. For, as we saw, $\mathcal{I}_{2A}$ assumes that there is already a fixed totality of *all* the arbitrary sets of numbers for the second-order quantifiers to run over. Given this semantics, an expression $\varphi(\mathsf{x})$ that embeds second-order quantifications does have a determinate sense. And, because we are assuming the existence of every possible extension, PA_2's generous comprehension principle is true on $\mathcal{I}_{2A}$. So, in sum, $\mathcal{I}_{2A}$ *intuitively makes all* PA_2*'s theorems true.*

Since PA_2 can quantify over arbitrary numerical sets, it can handle the standard sort of construction of the real numbers in terms of suitable infinite sets of numbers: we can therefore do a great deal of classical analysis in this theory (which is why the theory is sometimes just called *analysis*). But let that pass, for we are interested here in the *arithmetical* strength of PA_2 – i.e. the question of what it can prove by way of first-order arithmetical truths (true sentences of L_A). And the important headline news is this:

[6] So long as 'x' is free in $\varphi(\mathsf{x})$ and 'X' isn't, so we don't get into tangles. Incidentally, this theory is at least equally often called Z_2, the name given it in Hilbert and Bernays (1934).

[7] 'Whatever involves *all* of a collection must not be one of the collection' Russell (1908, p. 63). For discussion see Chihara (1973, esp. Ch. 1) and Potter (2000, esp. Ch. 5).

[8] The quotation is from Gödel (1933, p. 50); compare the remark on the supposed vicious circle in Ramsey (1925, p. 204).

Theorem 22.1 PA_2 *can prove every* L_A *wff that* PA *can prove, and some that* PA *can't prove.*

The first half of that theorem is, of course, trivial (because PA_2 includes Q, and PA_2's Induction Axiom together with its comprehension principle gives us all the first order instances of induction, i.e. gives us the other axioms of PA).

As for the second half, we just report e.g. that PA's canonical Gödel sentence G which is unprovable in PA is provable in PA_2.[9] Of course, PA_2 is not only 'neat' but 'nice' (assuming that the theory is consistent). Hence, while PA_2 *can* prove PA's Gödel sentence G, it *can't* prove its own canonical Gödel sentence G_2.[10]

22.5 Categoricity

PA and PA_2 are both intuitively sound theories, both incomplete. There are, however, deep differences between these two theories which we now need to highlight over the next two sections. To introduce them, we need some preliminary definitions (entirely familiar ones if you have done any serious logic at all).

i. Suppose T is a theory built in the language $L =_{\text{def}} \langle \mathcal{L}, \mathcal{I} \rangle$. Echoing what we just said in Section 22.1, an admissible (re)interpretation $\mathcal{J}$ for theory T is an interpretation of the syntax $\mathcal{L}$ which keeps the same interpretation of the logical vocabulary as is given by $\mathcal{I}$ but perhaps varies the interpretation of the non-logical vocabulary of T. If $\mathcal{J}$ makes all the axioms of a theory T true (and hence all T's theorems true too, assuming T's logic is a sound one), then we'll say that $\mathcal{J}$ is a *model* for T.

ii. Two admissible interpretations are *isomorphic* iff they structurally look exactly the same.

More carefully: $\mathcal{I}$ and $\mathcal{J}$ are isomorphic iff (i) there's a one-to-one correspondence f between the domain of $\mathcal{I}$ and the domain of $\mathcal{J}$ (see Section 2.3); (ii) if $\mathcal{I}$ assigns the object o as referent to the name a, then $\mathcal{J}$ assigns the corresponding object $f(o)$; (iii) if $\mathcal{I}$ assigns the set E as the extension to the predicate F, then $\mathcal{J}$ assigns the corresponding extension $f(E) =_{\text{def}} \{f(o) \mid o \in E\}$; and so on, and so forth. This means that Fa is true on $\mathcal{I}$ iff o is in E, which holds iff $f(o)$ is in $f(E)$ (since f is a one-one correspondence), and that holds just if Fa is true on $\mathcal{J}$. And the point obviously generalizes. In other words, a closed wff φ is

[9] We will see in Section 24.2 that even PA can prove $\mathsf{Con} \rightarrow \mathsf{G}$, where Con is a formal sentence that expresses the claim that PA is consistent: so certainly PA_2 can prove that too. But PA_2 is more than strong enough to prove Con. In fact it can prove e.g. the consistency of the richer theory ACA_0 which we'll meet in a moment; and if the stronger theory ACA_0 is consistent then the weaker theory PA which it extends must be consistent too: see Simpson (1991, Sec. VIII.1). Hence, by modus ponens, PA_2 can prove G.

[10] Note, by the way, that like any Gödel sentence built in the now familiar way, G_2 is a purely arithmetic Π_1 sentence that lacks second-order quantifiers.

true on interpretation $\mathcal{I}$ if and only if it is also true on any isomorphic interpretation $\mathcal{J}$.

iii. A theory is *categorical* iff it has models but all its models are isomorphic – i.e. the theory has just one model 'up to isomorphism', as they say.

And with that jargon to hand, we can state the first key contrast between PA and PA_2 like this:

Theorem 22.2 *Assuming both theories do have models,* PA *isn't categorical, but* PA_2 *is.*

Proof sketch: PA *isn't categorical* Assume PA has a model so is consistent. We then know that both of the expanded theories PA + R and PA + ¬R must also be consistent (where R is the undecidable Gödel-Rosser sentence). Hence, by the familiar theorem that any consistent first-order theory has a model, indeed a model with an enumerable domain, both these expanded theories have models. Since one of these models makes R true and the other makes ¬R true, these models can't be isomorphic replicas. But any model for PA + R and equally any model for PA + ¬R is a fortiori a model for plain PA. So PA has non-isomorphic models (with enumerable domains). In other words, PA is not categorical.[11] ⊠

Proof sketch: PA_2 *is categorical* Suppose $\mathcal{M}$ is a model for PA_2. $\mathcal{M}$ will need to pick out some object o to be the reference of '0', and to pick out some function ς which satisfies the successor axioms for 'S' to denote. And then, according to $\mathcal{M}$, the numerals $0, S0, SS0, SSS0, \ldots$ must pick out the objects $o, \varsigma o, \varsigma\varsigma o, \varsigma\varsigma\varsigma o, \ldots$. Let's abbreviate '$\varsigma\varsigma \ldots \varsigma o$' with m occurrences of 'ς' by '$\overline{m}$'. Then we can put it this way: according to $\mathcal{M}$, the numeral '$\overline{\mathsf{m}}$' picks out $\overline{m}$.

We'll denote the set of all the elements $\overline{m}$ by $\mathbb{M}$. And let's say that a model $\mathcal{M}$ is *slim* iff its domain is just the corresponding set $\mathbb{M}$ – i.e. iff its domain contains just the 'zero' element o, the 'successors' of this 'zero', *and nothing else*.

We now argue in two stages. (i) We'll show that any two *slim* models of PA_2 are isomorphic. Then (ii) we'll show that PA_2 can *only* have slim models. It will then follow that, if PA_2 has a model at all, it is a slim one, unique 'up to isomorphism' – a result essentially first shown by Richard Dedekind (1888).[12]

[11] For 'the familiar theorem', see any standard logic text, e.g. Mendelson (1997, p. 90, Prop. 2.17). But note also that, in order to show that PA isn't categorical, we don't in fact need to appeal to Gödelian incompleteness: the so-called 'Upward Löwenheim-Skolem Theorem' already tells us that any first-order theory like PA which has a model whose domain is the size of the natural numbers $\mathbb{N}$ has other, differently structured, models with bigger, non-enumerable, domains. What our Gödelian argument adds is that there are also non-isomorphic models of PA with enumerable domains. However, we haven't space in this book to pursue issues in model theory, so we can't here pause over the intriguing question of the fine structure of all the whacky, quite unintended, models that PA inevitably has.

For the general L-S theorem, see e.g. Mendelson (1997, p. 128), or perhaps more accessibly Bridge (1977, Ch. 4). For a wonderful exploration of the non-standard-but-enumerable models of arithmetic see the classic Kaye (1991).

[12] For a full-dress modern proof see e.g. Shapiro (1991, pp. 82–83). A technical note: it is crucial for Stage (ii) that the second-order quantifiers are required to run over the full collection

Stage (i) Suppose $\mathcal{M}$ is a slim model. We know that if $m \neq n$, then even BA $\vdash \overline{\mathsf{m}} \neq \overline{\mathsf{n}}$, and hence $\mathsf{PA}_2 \vdash \overline{\mathsf{m}} \neq \overline{\mathsf{n}}$. But $\mathcal{M}$ is a model and hence makes all PA_2's theorems true. So if $m \neq n$, then $\mathcal{M}$ must make $\overline{\mathsf{m}} \neq \overline{\mathsf{n}}$ true, and that requires $\overline{m} \neq \overline{n}$. Since m and n were arbitrary, it immediately follows that the objects $\overline{0}, \overline{1}, \overline{2}, \ldots$ must all be different. By hypothesis, $\mathcal{M}$'s domain $\mathbb{M}$ contains just those objects and nothing else; so $\mathbb{M}$ must look just like a copy of $\mathbb{N}$.

$\mathcal{M}$ will also pick out some function $+_{\mathcal{M}}$ for '+' to denote. It is easy to show that $+_{\mathcal{M}}$, when applied to the objects $\overline{m}$ and $\overline{n}$, also behaves 'just like addition': in other words, if $m + n = k$, then $\overline{m} +_{\mathcal{M}} \overline{n} = \overline{k}$. Similarly for multiplication. So again, there's only one way that 'addition' and 'multiplication' in a slim model can work: just like addition and multiplication of natural numbers.

So all the slim models must 'look the same', i.e. be isomorphic.

Stage (ii) Now consider the wff

$$\sigma(\mathsf{x}) =_{\text{def}} \forall \mathsf{Z}\{(\mathsf{Z0} \wedge \forall \mathsf{y}(\mathsf{Zy} \rightarrow \mathsf{ZSy})) \rightarrow \mathsf{Zx}\}.$$

PA_2's generous comprehension principle will apply to $\sigma(\mathsf{x})$ in particular. In other words, PA_2 treats $\sigma(\mathsf{x})$ as a kosher predicate expression that picks out a real property. So this property can figure in an instance of induction, and we have:

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

But $\sigma(\mathsf{0})$ and $\forall \mathsf{x}(\sigma(\mathsf{x}) \rightarrow \sigma(\mathsf{Sx}))$ hold trivially (check that!). So it is a PA_2 theorem that $\forall \mathsf{x}\sigma(\mathsf{x})$.

From which it follows that any model of PA_2, i.e. any interpretation $\mathcal{M}$ that makes all its theorems true, must make $\forall \mathsf{x}\sigma(\mathsf{x})$ true in particular. But, by definition, something satisfies $\sigma(\mathsf{x})$ on a given interpretation $\mathcal{M}$ just if it belongs to every subset Z of the domain which contains o and which if it contains x contains ςx (where o and ς are the zero and the successor function according to $\mathcal{M}$). Hence, in particular, something satisfies $\sigma(\mathsf{x})$ just if it belongs to the *smallest* subset of the domain which contains o and which if it contains x contains ςx. Hence something satisfies $\sigma(\mathsf{x})$ if it is the zero or one of its successors. So given $\mathcal{M}$ makes $\forall \mathsf{x}\sigma(\mathsf{x})$ true, every element of this model's domain must be either the zero or one of its successors. Therefore the model $\mathcal{M}$ is slim. ⊠

The paradigm slim model for PA_2 is, of course, $\mathcal{I}_{2A}$ (assuming that that interpretation *is* a model).

Note, incidentally, that the Stage (i) argument equally well shows that every slim model of first-order PA is isomorphic to the standard model $\mathcal{I}_A$. But we can't then run the Stage (ii) argument to show that PA is categorical and *only* has slim models; that's because the property of being a-zero-or-one-of-its-successors can't be expressed in L_A, and PA's Induction Schema can only deal with properties that can be expressed by L_A predicates.

of subsets of the domain. Versions of second-order arithmetic which lift that requirement – e.g. the theories which are in effect built in a two-sorted *first*-order language, explored in Simpson (1991) – aren't categorical.

22.6 Incompleteness and categoricity

(a) The fact that PA_2 is categorical and PA isn't entails another key difference between the first-order and second-order theory. Starting with the first-order case, suppose φ is some L_A truth that PA can't prove, e.g. a Gödel sentence like G. Then the axioms of PA don't semantically entail φ either. In standard symbols, if $\mathsf{PA} \nvdash \varphi$, then $\mathsf{PA} \nvDash \varphi$. That's because the built-in first-order deductive system is (sound and) complete; and hence for any φ, $\mathsf{PA} \vdash \varphi$ iff $\mathsf{PA} \vDash \varphi$.

Now for the contrasting second-order case: suppose φ is some L_{2A} truth which PA_2 can't prove, e.g. its true canonical Gödel sentence G_2. This time, however, PA_2 *will* still semantically entail φ. That's because we have:

> **Theorem 22.3** *Assuming $\mathcal{I}_{2A}$ is in fact a model for PA_2, PA_2 semantically entails all L_{2A}-truths.*

Proof sketch PA_2 semantically entails φ – in symbols, $\mathsf{PA}_2 \vDash_2 \varphi$ – if any admissible interpretation which makes the axioms of PA_2 true makes φ true. In other worlds, any model for PA_2 makes φ true.

However, we've just seen that PA_2 is a categorical theory. So given $\mathcal{I}_{2A}$ is a model for PA_2, *all* its models are isomorphic to $\mathcal{I}_{2A}$. But, trivially, if φ is true on $\mathcal{I}_{2A}$ (i.e. is an L_{2A}-truth), it will be true on all the isomorphic interpretations. It immediately follows that *all* interpretations which make the axioms of PA_2 true make φ true. So $\mathsf{PA}_2 \vDash_2 \varphi$, for any wff φ that is true on $\mathcal{I}_{2A}$. ⊠

Hence, in particular, although $\mathsf{PA}_2 \nvdash \mathsf{G}_2$, $\mathsf{PA}_2 \vDash_2 \mathsf{G}_2$. Which immediately reveals that *PA_2's deductive system isn't complete* – there are semantic entailments which it can't prove. And this isn't because we have rather dimly forgotten to give PA_2 enough logical deduction rules: the incompleteness can't be repaired, so long as we keep to a properly axiomatized logical system. For adding new logical axioms and/or inference rules to PA_2 will – so long as we stay consistent and decidably axiomatized – just give us a new theory which is still categorical and still subject to Gödel's incompleteness theorem. *There can be no sound and complete logical deductive system for second-order logical consequence.*

(b) In summary, then, while the claim 'PA settles all arithmetical truths' is false however we interpret it, the situation with the corresponding claim 'PA_2 settles all arithmetical truths' is more complex.

Assuming $\mathcal{I}_{2A}$ is a model for PA_2 (so the theory is consistent), the axioms of PA_2, interpreted as L_{2A} sentences, are enough to *semantically entail* all true sentences of L_{2A}. But the Gödel-Rosser Theorem tells us this formal deductive theory is not strong enough to *prove* all true L_{2A} sentences – and it can't be expanded to entail them all either, so long as the resulting formal theory remains consistent and properly axiomatized. Make the distinction between what is semantically entailed and what is proved, and we reconcile the apparent conflict between the implication of Dedekind's categoricity result ('PA_2 settles all the truths') and Gödelian incompleteness ('PA_2 leaves some truths undecided').

For vividness, let's put that symbolically. We'll use $\{\mathsf{PA}, \vdash\}$ to denote the set of theorems that follow in PA's formal proof system, and $\{\mathsf{PA}, \vDash\}$ to mean the set of sentences semantically entailed by PA's axioms (given the standard semantics of L_A). Similarly, we'll use $\{\mathsf{PA}_2, \vdash\}$ to mean the set of theorems that follow in PA_2's formal proof system, and $\{\mathsf{PA}_2, \vDash_2\}$ to mean the set of sentences semantically entailed by PA_2's axioms (given the 'full' semantics built into L_{2A}). Finally – as before, in Section 19.2 – we'll use $\mathcal{T}_A$ to denote the set of truths of L_A (True Basic Arithmetic); and we'll now use $\mathcal{T}_{2A}$, the set of truths of L_{2A} (True Second-Order Arithmetic). Then we can very perspicuously display the relations between these sets as follows, using $\subset$ to indicate strict containment (i.e. $X \subset Y$ when Y contains every member of X and more besides):

$$\{\mathsf{PA}, \vdash\} = \{\mathsf{PA}, \vDash\} \subset \mathcal{T}_A$$
$$\{\mathsf{PA}_2, \vdash\} \subset \{\mathsf{PA}_2, \vDash_2\} = \mathcal{T}_{2A}.$$

22.7 Another arithmetic

The main business of this chapter is done. But we'll add a couple of brief sections on related topics, one to link up with the theme of the following Interlude, the other to link up with remarks about speed-up at the end of the previous chapter.

As we saw, understanding the semantics $\mathcal{I}_{2A}$ built into PA_2 requires us to get our head around the infinitary concept of quantifying over all arbitrary sets of numbers. Suppose we want to restrict ourselves instead to the conceptual resources of non-infinitary pure arithmetic itself, and so adopt e.g. the interpretation $\mathcal{I}_{2a}$ (see the end of Section 22.1). Then we'll be interested in the corresponding weaker 'neat' arithmetic whose quantifiers in effect run over only those sets of numbers that we can pick out arithmetically.[13] Putting that formally, this involves requiring that any wff $\varphi(\mathsf{x})$ which we substitute into the Comprehension Scheme must already belong to L_A, i.e. be purely arithmetic and lack second-order quantifiers. The resulting neat axiomatized theory with this arithmetical comprehension principle is known in the trade as ACA_0.

Arguably, *this* theory doesn't go beyond what is given by our understanding of basic arithmetic (together with general logical ideas) – though despite its apparently unambitious character, we can *still* reconstruct a surprising amount of analysis and other applicable mathematics inside ACA_0. However, this time – by contrast with PA_2 – we get no new purely arithmetical truths:

Theorem 22.4 ACA_0 *is conservative over first-order* PA.

[13] 'Neat' is our informal term: there isn't a standard one. We should note, by the way, that not every interesting formal theory of second-order arithmetic is 'neat' in our sense: still, these neat theories – ordered by the increasing generosity of C, the class of wffs we can put into the Comprehension Scheme – do form a spine running through the class of second-order arithmetics. For an encyclopedic survey of theories of second-order arithmetic see Simpson (1991), whose first chapter gives a wonderfully helpful overview.

In other words, for any L_A sentence φ such that $\mathsf{ACA}_0 \vdash \varphi$, it is already the case that $\mathsf{PA} \vdash \varphi$.[14]

So we have the following suggestive contrast. We can derive more L_A sentences in PA_2 than in PA. But if we are to accept these formal derivations as *proofs* which give us reason to accept their conclusions, then we will need to accept the axioms of PA_2 as true. And to accept the axioms as true – including all those instances of Comprehension for formulae φ which quantify over subsets of the domain – will involve accepting infinitary ideas that go beyond those essential to a grasp of elementary arithmetic. By contrast, accepting the weaker formal theory ACA_0 doesn't involve more than a grasp of arithmetic together with some very general logical ideas; but then the theory doesn't give us any more basic arithmetic than PA. We'll return to this point in the next chapter.

22.8 Speed-up again

Finally, let's very briefly make a connection with Section 21.7, on the general topic of the length of proofs. We've said that ACA_0 proves just the same L_A wffs as PA; but interestingly it does *massively* speed up some proofs. Define

$$2 \star k = 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \quad \text{where the stack of 2s is } k \text{ high.}$$

Then Robert Solovay has shown that there is a family of L_A wffs $\varphi_0, \varphi_1, \varphi_2, \ldots$, such that (i) PA proves each φ_n; (ii) the PA proof of φ_n requires at least n bits, so in particular the PA proof of $\varphi_{2\star k}$ is at least $2 \star k$ bits long; but (iii) there is a constant c such that, for any k, there is an ACA_0 proof of $\varphi_{2\star k}$ less than ck^2 bits long. Hence, for all large enough values of n, there is an ACA_0 proof of φ_n which is *radically* shorter than the shortest proof for φ_n in PA.[15] And since any ACA_0 proof is also a PA_2 proof, the result carries over. There are PA theorems which have radically shorter proofs in PA_2.

Gödel himself noted this phenomenon long ago:

> [P]assing to the logic of the next higher order [e.g. moving from a first-order to a second-order setting] has the effect, not only of making provable certain propositions that were not provable before, but also of making it possible to shorten, by an extraordinary amount, infinitely many of the proofs already available. (Gödel, 1936, p. 397)

But a full proof of this Gödelian speed-up claim wasn't in fact published until Buss (1994).[16]

[14] For the proof, see Simpson (1991, Sec. IX.1).
[15] Solovay notes his result in http://www.cs.nyu.edu/pipermail/fom/2002-July/005680.html.
[16] For more, see Pudlák (1998).

23 Interlude: Incompleteness and Isaacson's conjecture

This Interlude discusses a couple of further questions about incompleteness that might well have occurred to you as you have been reading through recent chapters. But, given that the chapters since the last Interlude have been quite densely packed, we should probably begin with a quick review of where we've been.

23.1 Taking stock

Here's some headline news which is worth highlighting again:

1. First, we showed that the restriction of the First Theorem to p.r. axiomatized theories is in fact no real restriction. Appealing to a version of Craig's Theorem, we saw that the incompleteness result applies equally to any consistent axiomatized theory which contains Q (or, indeed, is otherwise p.r. adequate). (Section 19.1)

2. But our pivotal new result was Theorem 20.4, the general Diagonalization Lemma: if T is a nice theory and $\varphi(\mathsf{x})$ is *any* wff of its language with one free variable, then there is a 'fixed point' γ such that $T \vdash \gamma \leftrightarrow \varphi(\ulcorner\gamma\urcorner)$. And further, if $\varphi(\mathsf{x})$ is Π_1, then it has a Π_1 fixed point. (Section 20.5)

3. We then proved the rather easy Theorem 21.1: if γ is *any* fixed point for $\neg\mathsf{Prov}_T(\mathsf{x})$, then, if T is nice, $T \nvdash \gamma$, and if T is also ω-consistent, then $T \nvdash \neg\gamma$. Since the Diagonalization Lemma tells us that there *is* a fixed point for $\neg\mathsf{Prov}_T(\mathsf{x})$, that gives us the standard incompleteness theorem again. (Section 21.1)

4. We also proved the significantly messier Theorem 21.2: if T is a nice theory, and γ is *any* fixed point for $\neg\mathsf{RProv}_T(\mathsf{x})$, then $T \nvdash \gamma$ and $T \nvdash \neg\gamma$ – where the 'Rosserized' proof predicate $\mathsf{RProv}_T(\mathsf{x})$ says 'there's a proof of the wff with g.n. x, and no 'smaller' proof of its negation'. Since the Diagonalization Lemma also tells us that there is a fixed point for $\neg\mathsf{RProv}_T(\mathsf{x})$, we now get a proof of Theorem 19.6, the Gödel-Rosser Theorem. This allows us to drop talk of ω-consistency, and say simply: if T is a nice theory, then there is an L_A-sentence φ of Goldbach type such that neither $T \vdash \varphi$ nor $T \vdash \neg\varphi$. (Sections 19.3, 21.3)

This all reinforces our earlier remarks on incompletability. Suppose T is a nice theory. It's incomplete. Throw in some unprovable sentences as new axioms.

Then, by the Gödel-Rosser Theorem the resulting T^+ will still be incomplete, unless it stops being nice. But adding new axioms can't make a p.r. adequate theory any less adequate. So now we know the full price of T's becoming complete. Either (i) our theory ceases to be p.r. axiomatized because you've added too disordered a heap of new axioms, or (ii) it becomes flatly inconsistent. Outcome (i) is bad (for we now know that retreating to an axiomatized-but-not-p.r.-axiomatized theory won't let us escape the incompleteness results: a complete theory will have to stop being properly axiomatized at all). Outcome (ii) is worse. Keep avoiding those bad outcomes, and T is incompletable.

5. We then moved on to use the Diagonalization Lemma to prove Tarski's Theorem that, if the theory's language L is arithmetically rich enough to formulate a theory like Q, then L can't even *express* the numerical property of numbering a truth of L. (Section 21.5)

6. We also proved some results about the *length* of proofs. For example, if T is nice, then there are always some wffs which have relatively enormous proofs – take any p.r. function f at all, as fast growing as you like: there will be always be some wff with g.n. n whose proof has a g.n. greater than $f(n)$. (Section 21.7)

7. Finally, we explained how the incompleteness of even a nice theory like PA_2 is compatible with its categoricity (even though categoricity implies that – in one good sense – PA_2 *does* settle all arithmetical truths). In the second-order case, we crucially need to distinguish questions about what is *provable* in a theory from questions about what is *semantically entailed* by its axioms.

But, despite all that, our technical elaborations around and about the First Incompleteness Theorem still leave a number of unsettled questions. Here are two that might have occurred to you:

1. Back in Chapter 10, we said that PA is the benchmark first-order theory of basic arithmetic, and that – unlike Q, for example – it is not *obviously* incomplete. Since then, PA has turned out to be in fact incomplete, perhaps contrary to expectation. But our only specific examples of basic arithmetical truths that are unprovable-in-PA are Gödel sentences like G which are constructed using coding tricks; and, as we noted before, spelt out in all their detail, with all the abbreviations unpacked, these will be quite horribly long and messy sentences. Looked at purely as sentences of arithmetic, they have no *intrinsic* mathematical interest. Only someone who already knows about a particular, quite arbitrary, Gödel coding scheme will be in a position to recognize G as 'saying' that it is unprovable. Which raises the question: are there true sentences of basic arithmetic which *are* of intrinsic mathematical interest but which are not derivable in PA? Is PA, so to speak, *interestingly* incomplete; or are the gaps just Gödelian oddities?

2. If the ideas regimented by PA don't suffice to pin down the structure of the natural numbers, how come we all seem to arrive at a shared understanding of that structure (and apparently without making play with the infinitary ideas encapsulated in PA_2)? What further idea have we all got our heads around which pins down that structure?

In this Interlude, we will say something about these questions. However, the results we mention aren't taken up much in later chapters: so don't get bogged down, and do feel free to skim, or even to skip straight on to the discussion of the Second Theorem in the next chapter.

23.2 Goodstein's Theorem

(a) The first example of a more natural, non-Gödelian, arithmetical statement which is true, statable in the language of basic arithmetic, yet demonstrably not provable in PA, was found by Jeff Paris and Leo Harrington in 1977. However, the statement in question – a cunningly tweaked version of the so-called finite Ramsey Theorem – probably wouldn't strike you as *that* natural: and indeed it was, so to speak, cooked up for the occasion rather than being an interesting arithmetical truth known in advance. So let's concentrate instead on another arithmetical statement which was shown to be undecidable by PA a few years later by Paris and Laurence Kirby; it's a bit easier to explain, intriguing, and *was* already well known to be true.

(b) To set things up, we need an initial definition, the first of three:

i. The *pure base k representation* of n is the result of writing n as a sum of powers of k, then rewriting the various exponents of k themselves as sums of powers of k, then rewriting these new exponents as sums of powers of k, etc., ...

For example,

$$266 = 2^8 + 2^3 + 2^1.$$

So the pure base 2 representation of 266 is

$$266 = 2^{2^{(2^{2^0}+2^0)}} + 2^{(2^{2^0}+2^0)} + 2^{2^0}.$$

Similarly,

$$266 = 3^5 + 3^2 + 3^2 + 3^1 + 3^0 + 3^0.$$

And its pure base 3 representation is

$$266 = 3^{3^{3^0}+3^0+3^0} + 3^{3^0+3^0} + 3^{3^0+3^0} + 3^{3^0} + 3^0 + 3^0.$$

Now for our second definition:

ii. To evaluate the *bump function* $B_k(n)$, take the pure base k representation of n; bump up every k to $k+1$; and then subtract 1 from the resulting number.

Let's calculate, for example, $B_2(19)$:

Start with the pure base 2 representation of 19: $2^{2^{2^0}} + 2^{2^0} + 2^0$;
Bump up the base: $3^{3^{3^0}} + 3^{3^0} + 3^0$;
Subtract 1 to get $B_2(19) = 3^{3^{3^0}} + 3^{3^0} = 7625597484990$.

Here's our third definition:

iii. *The Goodstein sequence* for n is: n, $B_2(n)$, $B_3(B_2(n))$, $B_4(B_3(B_2(n)))$, etc.

In other words, start with n; then keep applying the next bump function to the last term in the sequence.

Let's give a couple of examples, the Goodstein sequences starting with 3 and 19 respectively. For brevity, we'll denote the k-th term of the sequence starting with m by 'm_k'.

$3_1 = 3$: i.e. $2^{2^0} + 2^0$
$3_2 = B_2(3_1) = 3^{3^0} + 3^0 - 1 = 3^{3^0}$
$3_3 = B_3(3_2) = 4^{4^0} - 1 = 4^0 + 4^0 + 4^0$
$3_4 = B_4(3_3) = 5^0 + 5^0 + 5^0 - 1 = 5^0 + 5^0$
$3_5 = B_5(3_4) = 6^0 + 6^0 - 1 = 6^0$
$3_6 = B_6(3_5) = 7^0 - 1 = 0$.

$19_1 = 19 = 2^{2^{2^0}} + 2^{2^0} + 2^0$
$19_2 = B_2(19_1) = 3^{3^{3^0}} + 3^{3^0} + 3^0 - 1 = 3^{3^{3^0}} + 3^{3^0} \approx 7 \cdot 10^{13}$
$19_3 = B_3(19_2) = 4^{4^{4^0}} + 4^{4^0} - 1 = 4^{4^{4^0}} + 4^0 + 4^0 + 4^0 \approx 1.3 \cdot 10^{154}$
$19_4 = B_4(19_3) = 5^{5^{5^0}} + 5^0 + 5^0 \approx 2 \cdot 10^{2184}$
$19_5 = B_5(19_4) = 6^{6^{6^0}} + 6^0 \approx 2.6 \cdot 10^{36305}$
$19_6 = B_5(19_5)$, etc., etc.

Which no doubt suggests that, while the Goodstein sequence for n might eventually hit zero for *very* small n, for later values of n the sequence, which evidently starts off wildly inflationary, must run away for ever.

(c) *But not so!* In his (1944), R. L. Goodstein showed that

Theorem 23.1 *For all n, the Goodstein sequence eventually terminates at zero.*

And very surprisingly, the proof is actually quite straightforward – or at least it is straightforward, if you know just a little of the theory of ordinal numbers. Let's quickly outline the proof:

Sketch of a proof sketch Take the Goodstein sequence for n. Render its k-th term into its pure base $k+1$ representation as in our examples above (with each sum presented in descending order of exponents). Now consider the parallel sequence that you get by going through and replacing each base number by ω (the first infinite ordinal). For example, the parallel sequence to the Goodstein sequence for 19 starts

$$\begin{aligned}
&\omega^{\omega^{\omega^{\omega^0}}} + \omega^{\omega^0} + \omega^0\\
&\omega^{\omega^{\omega^{\omega^0}}} + \omega^{\omega^0}\\
&\omega^{\omega^{\omega^{\omega^0}}} + \omega^0 + \omega^0 + \omega^0\\
&\omega^{\omega^{\omega^{\omega^0}}} + \omega^0 + \omega^0\\
&\omega^{\omega^{\omega^{\omega^0}}} + \omega^0\\
&\vdots
\end{aligned}$$

It isn't hard to show that this parallel sequence of ordinals will in every case be strictly decreasing.

But there can't be an infinite descending chain of such ordinals – that's a basic result in the theory of ordinals. Hence the ordinal sequence must terminate. And therefore the parallel Goodstein sequence for n must terminate too![1] ⊠

Don't worry at all, however, if you find that proof-sketch baffling: all you need to take away is the idea that Goodstein's Theorem *can* easily be proved, *if* we invoke ideas from the theory of infinite ordinals, i.e. if we invoke ideas that go beyond the basic arithmetic of finite numbers.

(d) Note next that the sequence-computing function $g(m,k) = m_k$ is evidently primitive recursive (to calculate the k-th value of the Goodstein sequence starting at m, we can write a program just using 'for' loops). Hence this two-place function is expressible in L_A by a three-place Σ_1 wff $\mathsf{S}(\mathsf{x},\mathsf{y},\mathsf{z})$.

Goodstein's theorem is therefore itself *expressible* in PA by the corresponding Π_2 sentence $\forall\mathsf{x}\exists\mathsf{y}\,\mathsf{S}(\mathsf{x},\mathsf{y},\mathsf{0})$. It is certainly not obvious, however, how we might go about *proving* Goodstein's theorem in PA, using only the arithmetic of finite numbers. Still, Theorem 21.7 has prepared us for the thought that there can be results which are very easily proved in stronger theories yet which only have horribly long proofs in PA. So might Goodstein's theorem be a case in point?

No. For here's the Kirby-Paris Theorem:[2]

> **Theorem 23.2** *Goodstein's Theorem is undecidable in* PA *(assuming* PA *is consistent).*

So even though the arithmetical proposition $\forall\mathsf{x}\exists\mathsf{y}\,\mathsf{S}(\mathsf{x},\mathsf{y},\mathsf{0})$ has an easy proof using just a light touch of the theory of ordinals, it can't be derived in PA itself.

[1] For a relatively gentle introduction to the ordinals, see e.g. Goldrei (1996). For a modern presentation of the theorem and its set-theoretic proof, see Potter (2004, pp. 212–218).

[2] For the original proof, see Kirby and Paris (1982); and for elaboration of the background Kaye (1991, Ch. 14). For a different method of proof, see Fairtlough and Wainer (1998).

For those who do know a bit about ordinals, we'll gesture at a proof of the Kirby-Paris Theorem, a proof that links it to a couple of key results which we'll be discussing later (again, *don't* worry if you find this mystifying!).

Sketch of a proof sketch Goodstein's Theorem depends on showing that there can't be an infinite decreasing chain of ordinals which are sums of powers of ω, i.e. there can't be an infinite decreasing chain of ordinals less than ε_0.[3] That depends in turn on supposing that *transfinite induction* up to ε_0 is sound.[4]

However, there are natural Gödel-numberings for the ordinals which are sums of powers of ω; so we can transmute claims about these ordinals into arithmetical claims about their numerical codes. And being able to prove Goodstein's theorem inside PA would be tantamount to PA's being able to handle (via coding) transfinite induction up to ε_0. But now we appeal to two future results. First, that kind of transfinite induction is in fact strong enough to prove the consistency of PA by Gentzen's method (see Section 24.7). So, if PA could prove Goodstein's theorem, it could prove its own consistency. But second, PA can't prove its own consistency, by Gödel's Second Theorem (see Section 24.3). So PA can't prove Goodstein's theorem. ⊠

So, in summary, the case of Goodstein's Theorem gives us an answer to the first question we posed at the beginning of this Interlude: there are indeed some non-Gödelian truths of basic arithmetic which have at least *some* intrinsic mathematical interest but which are provably independent of PA.

23.3 Isaacson's conjecture

The next key point to note is that the other known cases of mathematically interesting L_A truths which are provably independent of PA share an important feature with Goodstein's Theorem. *The demonstrations that they* are *L_A truths likewise use conceptual resources which go beyond those which are required for understanding the basic arithmetic of finite natural numbers.*

For example, we mentioned the Paris-Harrington result: proving it requires König's Lemma, which says that an infinite tree that only branches finitely at any point must have an infinite path through it.[5]

---

[3] $\varepsilon_0 = \omega^{\omega^{\omega^{\cdot^{\cdot^{\cdot}}}}}$ is the first ordinal that comes after all the sums of powers of ω.

[4] 'Transfinite induction up to ε_0'? One version of ordinary induction is this principle: suppose (i) 0 is F; and suppose (ii) if all numbers less than n are F, then n is F; then (iii) all numbers are F. Transfinite induction up to ε_0 is the parallel principle: suppose (i) 0 is F; and suppose (ii) if all ordinals less than α are F, then α is F (where α is a sum of powers of ω); then (iii) all ordinals which are sums of powers of ω are F.

[5] For details, see Kaye (1991, Ch. 14) again; the proofs of the variant Ramsey theorem and of its independence from PA were first given in Paris and Harrington (1977). For a gentle introduction see Kolata (1982) which also touches on Harvey Friedman's finite version of Kruskal's theorem, another truth which can be expressed in PA but is independent of it. Friedman's result is also discussed in Smoryński (1982) and Gallier (1991).

And – in a rather different way – appreciating the truth of undecidable Gödel sentences for PA also seems to involve conceptual abilities that go beyond a grasp of elementary operations on the finite numbers. Maybe in this case we don't need to get our head around highly infinitary ideas; but we surely have to be able to reflect on our own arithmetical theorizing in order to recognize e.g. that canonical Gödel sentences are true (see Section 27.8). We have to be able to make the move from (i) implicitly assuming in our unreflective mathematical practice that (say) every natural number has a unique successor to (ii) explicitly accepting that a certain theory which has that proposition as an axiom is sound/consistent. And this *is* a move, because knowing your way around the numbers doesn't in itself entail the capacity to be able to reflect on that ability.

Putting these points about the Gödelian and non-Gödelian cases together suggests an interesting speculation:

> *Isaacson's conjecture* If we are to give a rationally compelling proof of *any* true sentence of L_A which is independent of PA, then we will need to appeal to ideas that go beyond those which are constitutive of our understanding of basic arithmetic.[6]

If that's right, then PA in fact reaches as far into the truths of basic arithmetic as any properly axiomatized theory can reach, at least if it aims to encapsulate no more than what follows from our purely arithmetical knowledge.

But *is* the conjecture right? After all, it isn't exactly clear what is involved in 'purely arithmetical' knowledge. And you might initially have thought that it faces an immediate and very obvious challenge: for isn't there an evident way of going beyond first-order PA while still keeping within the confines of what is given to us in our understanding of elementary arithmetic – for can't we exploit our informal understanding of induction which seems to involve grasp of a *second-order* principle?

However, what we discovered about second-order arithmetics in the last chapter is in fact entirely in conformity with Isaacson's conjecture. To repeat, there are indeed L_A sentences which we can derive in PA_2 but which aren't derivable in PA. But if we are to accept these formal derivations as genuine *proofs* which do give us reason to accept their conclusions, then we must make sense of PA_2's

[6] Compare Daniel Isaacson's (1987), where he suggests that the truths of L_A that can't be proved in PA 'are such that there is no way that their truth can be perceived in purely arithmetical terms' (p. 203).

However, Isaacson goes further, adding – by way of explanation? – that 'via the phenomenon of coding [such truths] contain essentially hidden higher-order, or infinitary, concepts' (pp. 203–204). But that seems certainly wrong for the non-Gödelian cases. Take the unprovable wff $\forall x \exists y\, \mathsf{S}(x, y, 0)$, where S is as before the Σ_1 wff that expresses the p.r. function which computes the k-th member of the Goodstein sequence starting from m. In so far as there is any coding associated with this wff, it is the coding of the steps in an entirely finitary arithmetical computation. So although the *proof* of Goodstein's theorem involves infinitary concepts, the *content* of the theorem doesn't. But we won't pursue this point: for we can cleanly separate Isaacson's interesting conjecture from the rather contentious additional gloss he puts on it.

non-arithmetic, infinitary, idea of quantifying over arbitrary subsets of $\mathbb{N}$. By contrast, accepting e.g. the weaker formal theory ACA_0 doesn't seem to involve more than a grasp of arithmetic together with some very general logical ideas; but that theory doesn't give us any more basic arithmetic than PA does.

There is, to be sure, a whole range of neat (and not-so-neat) theories of intermediate strength worth considering, with comprehension axioms stronger than ACA_0's and weaker than PA_2's – and there is a lot of interesting work to be done to see, for example, exactly what strength of comprehension principle is required to prove various results like Goodstein's Theorem or the Gödel sentence for PA. However, the headline news is that these additional technicalities don't seem to change the basic picture as far as Isaacson's conjecture is concerned.

23.4 Ever upwards

Going *second-order*, we've noted, enables us to prove new *first-order* arithmetical sentences that we couldn't prove before. And we get the same phenomenon appearing again as we push on further up the hierarchy, and allow ourselves to talk not just of numbers and sets of numbers, but of sets of sets, sets of sets of sets, and so on. Embracing each new level up the hierarchy of types of sets allows us to prove some truths at lower levels which we couldn't prove before, including more ground-level L_A truths: richer theories can prove Gödel sentences for weaker theories (as PA_2 proves the canonical Gödel sentence for PA). Gödel himself remarked on this phenomenon even in his original paper:

> [I]t can be shown that the undecidable propositions constructed here become decidable whenever appropriate higher types are added. (Gödel, 1931, p. 181, fn. 48^a)[7]

And what about non-Gödelian cases? Are there intrinsically interesting arithmetical truths which are formally undecided by PA, also formally undecided by PA_2, and essentially *require* higher-order theories to prove them? Well, there is work by Harvey Friedman which aims to produce 'natural' arithmetical statements whose proof is supposed essentially to require the existence of 'large cardinals' – i.e. sets very high up the transfinite hierarchy. The jury is still out, however, both on the question of the naturalness of the results, and on the sense in which they really presuppose the existence of large cardinals.[8]

[7] If you know any set theory at all, then you'll know that the hierarchy of types of sets continues into the transfinite (so after all the finite types at levels $1, 2, 3, \ldots$, we take their union at level ω and keep on going through levels $\omega + 1$, $\omega + 2$ and up through the ordinals). And as we go further up, yet more ground-level arithmetical sentences become provable. Indeed, in his rather enigmatic footnote, Gödel suggests that the fact that 'the formation of ever higher types can be continued into the transfinite' – which means that there is an ever growing list of new arithmetic truths that become provable as we go up the hierarchy – is 'the true reason' for the incompleteness in any particular formal theory of arithmetic.

[8] Friedman's recent work is mostly announced in his many postings to the Foundations of Mathematics list: see http://www.cs.nyu.edu/mailman/listinfo/fom for a searchable archive. For

23.5 Ancestral arithmetic

(a) We have now responded to the first question we posed at the beginning of this Interlude. But our responses raise further issues: in particular, there's our second question, about the very business of understanding arithmetic.

As we said at the very outset (Section 1.1), it seems that anyone who comes to understand basic arithmetic gets to share an informal yet quite determinate conception of the structure of the natural number series ordered by the successor relation, and has an equally determinate conception of the operations of addition and multiplication for numbers. And in virtue of this shared grasp of these intuitive conceptions, it seems that we have a shared understanding of the idea of the standard, intended, interpretation for first-order PA. (We hope and believe, of course, that this standard interpretation is a model!)

However, assuming it is consistent, PA has non-isomorphic models. Which means that accepting its axioms as correct is quite compatible with understanding the theory as talking about some deviant, unintended model. So what more, beyond accepting PA's axioms as correct, *is* involved in understanding the standard structure of the natural numbers: what more is involved in fixing on the intended 'slim' model (whose domain comprises just a zero and its successors)?

It might have been tempting to conclude – given that the second-order Peano Arithmetic PA_2 *is* categorical and *does* pin down a unique structure (up to isomorphism) – that our everyday understanding must involve the sort of second-order ideas built into PA_2. But our discussion hasn't really favoured that suggestion. For that suggestion amounts to the idea that we need to understand the infinitary idea of quantifying over quite arbitrary subsets of the numbers in order to understand elementary arithmetic: and that surely overshoots. However, if we deploy instead some weaker and more tractable second-order ideas, like that of quantifying just over arithmetical properties, we end up with a theory which in crucial ways is no stronger than PA. So what to do?

(b) To pin down the intended 'slim' model of PA, we need the idea that the numbers comprise zero, the successors of zero (all different, and never circling round to zero again), *and nothing else*. In other words, the numbers are what you can reach by repeated moves from a number to its successor.

Here's a similar idea at work. We familiarly define the class of wffs of an formal language by specifying a class of atomic wffs, and then saying e.g. that any wffs you can build up from these using connectives and quantifiers are wffs, and nothing else is a wff. So in this case we are saying that the wffs are what you can get by repeated applications of the wff-building operations.

Or take an even simpler case, the notion of an ancestor. An ancestor is someone you can reach by repeated moves back from a person to one of their parents. Now, to be sure, we *can* define the idea of ancestor from the idea of a parent

a programmatic outline of his overall approach, see Friedman (2000). For a critical response, see Feferman (2000).

by invoking second-order ideas (i.e., in this case, by talking not just of people but of sets of people). Thus someone is one of my ancestors if they belong to every set of people which includes me and which, if it includes N, also includes N's parents. Further, we can't define the idea of an ancestor from the idea of a parent using just first-order quantifiers, identity and the connectives.[9]

However, although we *can* define the idea of an ancestor from the idea of a parent in second-order terms, and *can't* do it first-order terms, it surely doesn't follow that the child who so easily grasps the concept *ancestor* must already in effect be quantifying over arbitrary sets of people.

Likewise we can define the idea of a wff in second-order terms. For example, an expression is a wff of L_A iff it belongs to every collection of expressions which contains certain atoms, and which, if it contains φ and ψ, contains $\neg\varphi$, $(\varphi \wedge \psi)$, $\forall\xi\varphi$, etc. But the availability of this sort of definition again doesn't show that the humdrum notion of an L_A-wff – something that we need to grasp to understand the Induction Schema of first-order PA – is already essentially second-order. Here's a comparison. We can define identity in second-order terms by putting $\mathsf{x} = \mathsf{y} =_{\text{def}} \forall\mathsf{X}(\mathsf{Xx} \leftrightarrow \mathsf{Xy})$; but that doesn't show that understanding identity is to be explicated in terms of understanding quantification over arbitrary subsets of the domain. It can't show that, because understanding the idea of arbitrary sets of objects of some kind presupposes an understanding of what makes for an object, and that involves an understanding of what makes candidates the same or different objects, i.e. it already involves an understanding of identity.

So, in sum, to pin down the intended domain of numbers (to keep it slim, banning extraneous objects) requires deploying the general idea of something being one of the things we can reach by repeating a certain operation indefinitely often, the same idea that is involved in specifying what it is to be a wff or an ancestor. And the suggestion is that, although this idea isn't definable in first-order terms, it is *not* second-order in the full sense either.

(c) So the child learns that her parents have parents, and that they have parents too. In sepia tones, her great-grandparents have parents in their turn. And she learns that other children have parents and grandparents too. The penny drops: she realizes that in each case you can keep on going. And the child gets the idea of an ancestor, i.e. the idea of being someone who turns up eventually as you trace people back through their parents. Similarly, she learns to count; the hundreds are followed by the thousands, and the tens of thousands, and the hundreds of thousands and the millions. Again the penny drops: you can keep on going. And she gets the idea of being a natural number, i.e. of turning up eventually as you keep on moving to the next number.

Generalizing, then, these are cases where the child moves from a grasp of a

[9] For a proper proof, see Shapiro (1991, pp. 99–100). But the following thought is suggestive. If R stands for the relation of *being a parent to*, then we could express 'm is an ancestor of n' by $\mathsf{Rmn} \vee \exists\mathsf{x}(\mathsf{Rmx} \wedge \mathsf{Rxn}) \vee \exists\mathsf{x}\exists\mathsf{y}(\mathsf{Rmx} \wedge \mathsf{Rxy} \wedge \mathsf{Ryn}) \vee \exists\mathsf{x}\exists\mathsf{y}\exists\mathsf{z}(\mathsf{Rmx} \wedge \mathsf{Rxy} \wedge \mathsf{Ryz} \wedge \mathsf{Rzn}) \vee \ldots$, if we were allowed infinitely long sentences. But a first-order language doesn't allow unending disjunctions, nor in general does it allow us to construct equivalent wffs.

relation to a grasp of the *ancestral* of that relation – where the ancestral R^* of a relation R is that relation such that R^*ab holds just when there is an indefinitely long chain of R-related things between a and b. Now, we don't want to be multiplying conceptual primitives unnecessarily. But the concept of the ancestral of a relation doesn't seem a bad candidate for being a fundamental logical idea: grasping this concept seems a distinctive level of cognitive achievement. So what happens if we try to build *this* idea into a formal theory of arithmetic which is otherwise basically first order? – which is an old question that goes back at least to R. M. Martin and John Myhill over fifty years ago.[10]

(d) To begin, we need to expand our first-order logical vocabulary with an operator – we'll symbolize it with a star – which attaches to two-place expressions $\varphi(\mathsf{x},\mathsf{y})$: $\varphi^*(\mathsf{x},\mathsf{y})$ is to be interpreted as expressing the ancestral of the original relation expressed by $\varphi(\mathsf{x},\mathsf{y})$.[11]

Suppose that we augment the language L_A with such an operator; and for brevity we'll write the result of applying the star operator to $\mathsf{Sx} = \mathsf{y}$ as $\mathsf{S}^*\mathsf{xy}$. So consider then what happens when you take the familiar axioms of PA but add in the new axiom

S. $\forall \mathsf{x}(\mathsf{x} = 0 \vee \mathsf{S}^*0\mathsf{x})$

which says – as we want – that every number is zero or one of its successors. Any interpretation of this expanded set of axioms which respects the fixed logical meaning of the ancestral-forming operator evidently must be slim. So, by the argument for Theorem 22.2, a theory with these axioms will be categorical. Let's define, then, the related semantic entailment relation $\Gamma \vDash^* \varphi$, which obtains if every interpretation which makes all of Γ true makes φ true – where we are now generalizing just over interpretations which give the star operator its intended meaning (and otherwise treats the logical vocabulary standardly). Then, because of categoricity, our expanded axioms semantically entail any true sentence of the expanded language, and hence any true L_A sentence.

There can't, however, be a complete axiomatization of this 'ancestral arithmetic'. The argument is as before. Take any formally axiomatized theory A^* which extends PA plus (S) by adding rules for the star operator. The incompleteness theorem still applies (assuming consistency), so there will be an unprovable-yet-true L_A sentence G^* for this theory. In other words, $\mathsf{A}^* \nvdash \mathsf{G}^*$, although $\mathsf{A}^* \vDash^* \mathsf{G}^*$. So our theory's deductive system can't be complete.

However, there can of course be *partial* axiomatizations of ancestral arithmetic. We can lay down various rules for handling the ancestral operator. Suppose, for brevity, that we write $\mathsf{H}(\psi,\varphi)$ as short for $\forall \mathsf{x}\forall \mathsf{y}((\psi(\mathsf{x}) \wedge \varphi(\mathsf{x},\mathsf{y})) \rightarrow \psi(\mathsf{y}))$ – i.e. the property expressed by ψ is *hereditary* with respect to the relation φ

[10] See Martin (1943) and the follow-up note (1949) where Martin urges that his construction is 'nominalistic', i.e. doesn't commit you to the existence of sets. This work was then developed in the rather more accessible Myhill (1952).

[11] We are being forgivably careless about the syntactic details here.

(i.e. is passed down a chain linked by φ). Then Myhill's proposed axioms for the star operator are tantamount to the following schematic rules:

From $\varphi(\mathsf{a}, \mathsf{b})$ infer $\varphi^*(\mathsf{a}, \mathsf{b})$.
From $\varphi^*(\mathsf{a}, \mathsf{b})$, $\varphi(\mathsf{b}, \mathsf{c})$ infer $\varphi^*(\mathsf{a}, \mathsf{c})$.
From $\mathsf{H}(\psi, \varphi)$, infer $\mathsf{H}(\psi, \varphi^*)$.

These first two rules are 'introduction' rules for the star operator; and the third rule is easily seen to be equivalent to the following 'elimination' rule:

From $\varphi^*(\mathsf{a}, \mathsf{b})$ infer $\{\psi(\mathsf{a}) \wedge \mathsf{H}(\psi, \varphi)\} \rightarrow \psi(\mathsf{b})$.

This is a kind of generalized inductive principle which says that given that b is a φ descendant of a then if a has some property which is passed down from one thing to another if they are φ related, then b has that property too. And taking the particular case where $\varphi(\mathsf{x}, \mathsf{y})$ is $\mathsf{Sx} = \mathsf{y}$, having this rule will enable us to derive all the instances of the familiar first-order induction schema.

So let's now briefly consider the formal system PA^* which extends Q by adding our new axiom (S) plus Myhill's rules for handling the ancestral operator. The obvious next question is: what is the deductive power of this system? Is this another case like PA_2 where we also only had a partial axiomatization of the relevant semantic relation, though PA_2 could prove more L_A sentences than PA. Or is PA^* another extension of PA like ACA_0, which is conservative over PA?

If the first case held, then we'd have a very interesting challenge to Isaacson's conjecture. For the ancestral arithmetic PA^* is arguably within the conceptual reach of someone who has fully understood basic arithmetic; and so, if we could use it to prove new sentences of basic arithmetic not provable in PA, then Isaacson's conjecture would fall. But in fact, the issue doesn't arise:

Theorem 23.3 PA^* *is conservative over* PA.

In other words, for any L_A sentence ψ such that $\mathsf{PA}^* \vdash \psi$, it is already the case that $\mathsf{PA} \vdash \psi$.

Proof sketch Recall that we can express facts about sequences of numbers in PA by using a β-function (see Section 13.4). So suppose R is some relation. Then

A. R^*ab is true just so long as, for some x, there is a sequence of numbers $k_0, k_1, \ldots, k_x$ such that: $k_0 = a$, and if $u < x$ then Rk_uk_{Su}, and $k_x = b$.

Using a three-place β-function, that means

B. R^*ab is true iff for some x, there is a pair c, d such that: $\beta(c, d, 0) = a$, and if $u < x$ then $R(\beta(c, d, u), \beta(c, d, Su))$, and $\beta(c, d, x) = b$.

So consider the following definition:

C. $\varphi^{**}(\mathsf{a}, \mathsf{b}) =_{\text{def}} \exists\mathsf{x}\exists\mathsf{c}\exists\mathsf{d}\{\mathsf{B}(\mathsf{c}, \mathsf{d}, 0, \mathsf{a}) \wedge$
$(\forall\mathsf{u} \leq \mathsf{x})[\mathsf{u} \neq \mathsf{x} \rightarrow \exists\mathsf{v}\exists\mathsf{w}\{(\mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{u}, \mathsf{v}) \wedge \mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{Su}, \mathsf{w})) \wedge \varphi(\mathsf{v}, \mathsf{w})\}] \wedge$
$\mathsf{B}(\mathsf{c}, \mathsf{d}, \mathsf{x}, \mathsf{b})\}$

where B captures e.g. the now familiar three-place Gödelian β-function.

It is easy to check that the Myhill inference rules for single-starred φ^* apply equally to our defined double-starred construct φ^{**} in PA (that's essentially because the moves are then valid semantic entailments within PA, and the theory's deductive system is complete). And the double-starred analogue of axiom (S) is also a theorem of PA. So corresponding to any proof involving starred wffs in PA* there is an exactly parallel proof in plain PA involving double-starred wffs. Hence in particular, any proof using starred wffs whose conclusion is a pure (unstarred) L_A wff in PA* will have a parallel proof in plain PA going via double-starred wffs. Which establishes what we needed to show.[12] ⊠

(e) Let's summarize this last section. Our everyday understanding of basic arithmetic pins down a unique structure for the natural numbers, at least up to isomorphism. Hence our grasp of basic arithmetic involves more than is captured by first-order PA. But what more? It is enough that we have the idea that the natural numbers are zero and its successors and nothing else. And getting our head round this idea, we suggested, involves the general idea of the ancestral: the numbers are what stand in the ancestral of the successor relation to zero.

Now, the ancestral of a relation can be defined in second-order terms, but it seems overkill to suppose that our understanding of ordinary school arithmetic is essentially second-order. Why not treat the operation that takes a relation to its ancestral to be a (relatively unmysterious, even though not purely first-order) logical primitive? If we do, we can construct a theory PA* which naturally extends PA in a way that arguably still reflects our everyday understanding of arithmetic. And PA* has the semantic property of categoricity – it pins down the structure of the natural numbers in exactly the way we want.

Since PA* is still properly axiomatized, however, we know that it will be incomplete (assuming it is consistent). But we might have hoped that it would at least have proved more than PA: but not so. PA* is deductively conservative over PA for L_A sentences: so we can't in fact use this expanded theory to deduce new truths of basic arithmetic that are left unsettled by PA.

Hence, to put it the other way around, it seems that if we are to come up with formal proofs of L_A truths unsettled by PA, then we'll have to deploy premisses and/or logical apparatus that go beyond PA* (or simple variants). Which arguably implies that we'll need to invoke ideas which go beyond those essential to our ordinary understanding of basic arithmetic – for the idea that all the numbers can be reached from zero by repeatedly adding one, i.e. the idea that all numbers are related to zero by the ancestral of the successor relation, is very plausibly at the limit of what is *necessary* to ground a grasp of the natural number structure and arithmetic concepts which can be defined over it. Which gives us Isaacson's conjecture again.

[12] Thanks to Andreas Blass and Aatu Koskensilta for discussion of this. The argument would seem evidently to generalize to any natural first-order variant of PA*. For something on richer, second-order, ancestral logics, see Heck (2007).

24 Gödel's Second Theorem for PA

We now, at long last, turn to considering the *Second* Incompleteness Theorem for PA.

We worked up to the First Theorem very slowly, spending a number of chapters proving various preliminary technical results before eventually taking the wraps off the main proofs in Chapters 16 and 17. But things seem to go rather more smoothly and accessibly if we approach the Second Theorem the other way about, working backwards from the target Theorem to proofs of the technical results needed to demonstrate it. So in this chapter, we simply *assume* a background technical result about PA which we will call the 'Formalized First Theorem': we then show that it immediately yields the Second Theorem for PA when combined with Theorem 20.2.

In the next chapter, we show that the Formalized First Theorem and hence the Second Theorem can similarly be derived in any arithmetic theory T for which certain 'derivability conditions' hold (or rather, hold in addition to the Diagonalization Lemma). Then in Chapter 26 we finally dig down to discover what it takes for those derivability conditions to obtain.

24.1 Defining Con

We begin with four reminders, and then motivate a pair of new definitions:

1. Recall, $Prf(m, n)$ holds when m is the super g.n. of a PA-proof of the wff with g.n. n. And we defined $Prov(n)$ to be true just when n is the g.n. of a PA theorem, i.e. just when $\exists m\, Prf(m, n)$. Thus, $Prov(\ulcorner\varphi\urcorner)$ iff $\mathsf{PA} \vdash \varphi$. (See Sections 15.2, 15.9 and 20.1.)

2. Recall, we used $\mathsf{Prf}(\mathsf{x}, \mathsf{y})$ as an abbreviation for a Σ_1 wff of PA's language that canonically captures the relation *Prf* by perspicuously recapitulating its p.r. definition. (Again, see Section 20.1: note that previously we didn't actually make much use of the fact that $\mathsf{Prf}(\mathsf{x}, \mathsf{y})$ *canonically* captures *Prf*. But now this becomes rather crucial – as we will explain, particularly in Section 26.2.)

3. Recall, we went on to define $\mathsf{Prov}(\mathsf{x}) =_{\text{def}} \exists \mathsf{v}\, \mathsf{Prf}(\mathsf{v}, \mathsf{x})$. This complex predicate is also Σ_1, and it expresses the numerical property *Prov*. Hence $\mathsf{Prov}(\ulcorner\varphi\urcorner)$ is true iff $\mathsf{PA} \vdash \varphi$. (See Section 20.1.)

4. Recall, finally, that we showed that $\mathsf{PA} \vdash \mathsf{G} \leftrightarrow \neg\mathsf{Prov}(\ulcorner\mathsf{G}\urcorner)$. (That was Theorem 20.2.)

5. Now to move on to our new definitions. Suppose that we have set up the first-order logic of PA using the familiar absurdity constant '$\bot$'. Then, of course, PA is consistent if and only if $\mathsf{PA} \nvdash \bot$.

   Suppose on the other hand that the absurdity constant isn't built into PA's logic. But PA's Axiom 1 immediately proves the wff $0 \neq 1$; so if PA also proved $0 = 1$, it would be inconsistent. And conversely, since PA has a classical logic, if it is inconsistent we can derive anything, including $0 = 1$. So, when '$\bot$' isn't built in, let's put $\bot =_{\text{def}} 0 = 1$. Again, PA is consistent if and only if $\mathsf{PA} \nvdash \bot$.

6. Henceforth, then, we'll take the absurdity constant '$\bot$' to be available in PA, either built in or by definition (and similarly in other theories).

   But PA is consistent, i.e. $\mathsf{PA} \nvdash \bot$, just when it isn't the case that $Prov(\ulcorner \bot \urcorner)$, i.e. just when $\neg\mathsf{Prov}(\ulcorner \bot \urcorner)$ is true. So that motivates the following abbreviation:

   $$\mathsf{Con} =_{\text{def}} \neg\mathsf{Prov}(\ulcorner \bot \urcorner).$$

   Being the negation of a Σ_1 sentence, Con is Π_1.

So, the sentence Con is true if and only if PA is consistent. But more than that, given the background Gödel coding scheme in play, we can immediately see, without need for any further argument, that Con is constructed in such as way as to make it true if and only if PA is consistent. Or, as we might perhaps say, the sentence Con *indirectly says* that PA is consistent.[1]

24.2 The Formalized First Theorem in PA

In Section 16.5 we proved the following (it's the easier half of the First Theorem):

> If PA is consistent, then G is not provable in PA.

We now know that one way of representing the antecedent of this conditional in L_A is by the formal wff we are abbreviating as Con, while the consequent can of course be represented by $\neg\mathsf{Prov}(\ulcorner \mathsf{G} \urcorner)$. So, in sum, the wff

$$\mathsf{Con} \rightarrow \neg\mathsf{Prov}(\ulcorner \mathsf{G} \urcorner)$$

expresses one half of the incompleteness theorem for PA *inside* PA *itself*.

But that point by itself isn't particularly exciting. The novel and interesting claim is the next one, call it the *Formalized First Theorem*:

F. $\mathsf{PA} \vdash \mathsf{Con} \rightarrow \neg\mathsf{Prov}(\ulcorner \mathsf{G} \urcorner)$.

In other words, (half of) the incompleteness theorem is not merely *expressible* in PA but is actually *provable* in PA too.

[1] Compare our remarks about what G 'indirectly says' at the end of Section 16.8.

Of course, we've seen this sort of thing before. Remember how we initially constructed G in Section 16.2, and then immediately noted that G is true if and only if it is unprovable. That informally derived biconditional (which we reached, so to be speak, while looking at PA 'from the outside') can readily be expressed in the language of PA, by the wff $\mathsf{G} \leftrightarrow \neg\mathsf{Prov}(\ulcorner\mathsf{G}\urcorner)$. And, as we've just reminded ourselves, Theorem 20.2 tells us that this biconditional can be formally *proved* inside PA:

D. $\mathsf{PA} \vdash \mathsf{G} \leftrightarrow \neg\mathsf{Prov}(\ulcorner\mathsf{G}\urcorner)$.

We now have a very similar situation. Informal reasoning (looking at PA 'from the outside') leads to Gödel's result that PA's consistency entails G's unprovability. And now we are saying that this result can in fact be formally derived within PA itself: so (F) holds.

Gödel doesn't actually *prove* (F) or anything like it in his paper.[2] He just invites us to observe the following:

> All notions defined or statements proved [in establishing the First Theorem] are also expressible or provable in P [the formal system Gödel is working with]. For throughout, we have used only the methods of definition and proof that are customary in classical mathematics, as they are formalizable in P. (Gödel, 1931, p. 193)

Gödel could very confidently assert this because P, recall, is his version of Russell and Whitehead's theory of types; it is a higher-order theory which is a lot richer than either first-order PA or second-order PA_2, and which was already well known to be sufficient to formalize great swathes of mathematics. So, agreed, it is entirely plausible that P has the resources to formalize the very straightforward mathematical reasoning that leads to the First Theorem for P.

It isn't so obvious, however, that all the reasoning needed for the proof of the First Theorem for PA can be formally reflected in a relatively low-power theory like PA itself. Checking that we can formalize the proof inside PA (indeed, inside weak subsystems of PA) requires some hard work. But let's take it for the moment that the work has been done – that's the big technical assumption on which the rest of this chapter proceeds. Then we will have arrived at the result (F), the Formalized First Theorem.

24.3 The Second Theorem for PA

Suppose now (for reductio) that

1. $\mathsf{PA} \vdash \mathsf{Con}$.

Then, given the Formalized First Theorem, modus ponens yields

[2] It seems that he intended to make good the deficit in a never-written Part II of his paper.

2. $\mathsf{PA} \vdash \neg\mathsf{Prov}(\ulcorner \mathsf{G} \urcorner)$.

But (D) tells us that $\neg\mathsf{Prov}(\ulcorner \mathsf{G} \urcorner)$ and G are provably equivalent in PA. Whence

3. $\mathsf{PA} \vdash \mathsf{G}$.

However, that contradicts the First Theorem. Therefore our supposition (1) must be false, unless PA is inconsistent.

Hence, still assuming we *can* establish the Formalized First Theorem, that shows

Theorem 24.1 *If* PA *is consistent,* $\mathsf{PA} \nvdash \mathsf{Con}$.

Call this result Gödel's *Second Incompleteness Theorem* for PA. If we assume that the axioms of PA are true on the standard interpretation and hence all its theorems are true (so PA *is* consistent), Con will be another true-but-unprovable Π_1 wff.

24.4 On ω-incompleteness and ω-consistency again

(a) Assume PA *is* consistent. Then $\bot$ isn't a theorem. So no number is the super g.n. of a proof of $\bot$ – i.e. for all n, it isn't the case that $Prf(n, \ulcorner \bot \urcorner)$. Since Prf captures Prf, we therefore have

1. for any n, $\mathsf{PA} \vdash \neg\mathsf{Prf}(\mathsf{n}, \ulcorner \bot \urcorner)$.

Now, if we *could* prove Con then – unpacking the abbreviation – we'd have

2. $\mathsf{PA} \vdash \forall \mathsf{v} \neg\mathsf{Prf}(\mathsf{v}, \ulcorner \bot \urcorner)$.

The unprovability of Con means, however, that we *can't* get from (1) to (2). So this is another example of PA's *ω-incompleteness* (see Section 16.6). In fact, since Con is Π_1, i.e. of Goldbach type in the sense of Section 16.4, this is another example of the failure of ω-completeness for wffs of Goldbach type.

The situation, then, is this. We know that – once we have introduced Gödel numbering – lots of arithmetical truths coding facts about provability-in-PA can themselves be proved in PA. And in so far as we might originally have expected PA to be ω-complete, we might reasonably have expected Con in particular to be provable. But now that we know that examples of ω-incompleteness are endemic in otherwise nice theories, Theorem 24.1 is perhaps not such a surprise.

(b) Suppose that $\mathsf{PA} \vdash \neg\mathsf{Con}$. In other words, suppose that $\mathsf{PA} \vdash \exists \mathsf{v} \mathsf{Prf}(\mathsf{v}, \ulcorner \bot \urcorner)$. This, given (1) above, would make PA ω-inconsistent. Which immediately gives us the following very easy companion result to Theorem 24.1 (this one *doesn't* depend on the Formalized First Theorem):

Theorem 24.2 *If* PA *is ω-consistent, then* $\mathsf{PA} \nvdash \neg\mathsf{Con}$.

24.5 How should we interpret the Second Theorem?

So much for the initial technical results. In the rest of this chapter we say some introductory things about how we should interpret the Second Theorem for PA and about why the Theorem might be interesting. You might want to skim lightly on a first reading.

(a) Looking at it one way, all we've actually done so far in this chapter is to locate another wff which exemplifies Gödel's *First* Incompleteness Theorem as applied to PA. That theorem (Theorem 16.5) states that there is an L_A-sentence φ of Goldbach type such that, if PA is consistent then $\mathsf{PA} \nvdash \varphi$, and if PA is ω-consistent then $\mathsf{PA} \nvdash \neg\varphi$. Our first sample case of such an undecidable φ was PA's canonical Gödel sentence G: now we've found another such φ, namely the sentence Con. Which raises the question: what is the special significance of locating this new undecidable sentence?[3]

Gödel interprets his generalized version of the Second Theorem in the following terms:

> [For a suitable consistent theory κ] the sentential formula stating that κ is consistent is not κ-provable; in particular, the consistency of P is not provable in P, provided P is consistent. (Gödel, 1931, p. 193)

So the analogous gloss on our Second Theorem for PA would be: *the consistency of* PA *is not* PA*-provable, assuming* PA is *consistent.*

But is that quite right? For when we introduced it, we said that Con is *one* natural way of 'indirectly saying' that PA is consistent. But we haven't shown that it is the *only* way of expressing PA's consistency in L_A. So although PA can't prove Con, the question remains: couldn't it still prove some *other* sentence which states that PA is consistent?

(b) Here are two obvious alternative ways of expressing PA's consistency in L_A. First, we could define

$$\mathsf{Con}' =_{\text{def}} \neg\exists \mathsf{x}(\mathsf{Prov}(\mathsf{x}) \wedge \mathsf{Prov}(\dot{\neg}\mathsf{x})).$$

Here, '$\dot{\neg}$x' expresses the p.r. function that maps the g.n. of a wff φ to the g.n. of its negation $\neg\varphi$ (see Section 21.3, fn. 4 for more explanation of our dotting notation). Second, relying on the fact that inconsistent classical theories 'prove' every sentence, we could define

$$\mathsf{Con}'' =_{\text{def}} \exists \mathsf{x}(\mathsf{Sent}(\mathsf{x}) \wedge \neg\mathsf{Prov}(\mathsf{x})),$$

[3]Note that Con, like G, unpacks into pure L_A as just another horribly complicated arithmetical sentence, whose cumbersome details will be contingent on the vagaries of our choice of coding. This unpacked L_A sentence will have no more intrinsic mathematical interest than the unpacked version of G. It is only when we look at it through the lens of coding that Con becomes interesting.

where $\mathsf{Sent}(x)$ captures the p.r. property of being a closed wff of PA's language (this happens to be Gödel's own preferred type of consistency sentence).

Still, as you'd probably expect, we can show without too much effort that $\mathsf{PA} \vdash \mathsf{Con} \leftrightarrow \mathsf{Con}'$, and $\mathsf{PA} \vdash \mathsf{Con} \leftrightarrow \mathsf{Con}''$. Likewise for the other most obvious ways of expressing PA's consistency in L_A. And being provably equivalent to Con, these alternative consistency sentences are of course all equally unprovable in PA. It is therefore reasonable to think of Con as the *canonical* consistency statement for PA, unique up to provable equivalence.

(c) But could there perhaps be other, more oblique and less obvious, ways of expressing PA's consistency in L_A? Well, there is a genuine issue here. But let's not get tangled up in this further question now: instead, we will return to the topic in Section 27.2. For the moment we'll just announce that the headline news is this: *if we care about informative consistency proofs, the canonical consistency statement* Con *(or an equivalent) is what we'll want to prove.* And while we might well have expected an arithmetically rich consistent theory like PA to 'know about' its own consistency (when we code up that claim), it can't. In fact – as the next chapter's generalized version of the Second Theorem shows – if *any* such theory 'proves' its own consistency, it must be *in*consistent.

24.6 How interesting is the Second Theorem for PA?

You might well think: 'OK, so we can't derive Con in PA: but that fact is of course no evidence at all *against* PA's consistency, since we already know from the First Theorem that lots of true claims about provability are underivable in PA. While if, *per impossibile*, we could have given a PA proof of Con, that wouldn't have given us any special evidence *for* PA's consistency – we could simply reflect that even if PA were inconsistent we'd still be able to derive Con, since we can derive *anything* in an inconsistent theory! Hence the derivability or otherwise of a canonical statement of PA's consistency inside PA itself can't show us a great deal.'

But, on reflection, the Theorem *does* yield three plainly important and substantial corollaries. The Theorem tells us (a) that *even* PA isn't enough to derive the consistency of PA, so we certainly can't derive the consistency of PA using a *weaker* theory. It tells us (b) that PA isn't enough to derive the consistency of *even* PA, so we certainly can't use PA to demonstrate the consistency of a *stronger* theory. And it tells us (c) that if we *are* going to produce any interestingly informative consistency proof for PA then, since a weaker theory isn't up to the job and using a stronger theory which fully subsumes PA would be question-begging, we'll need to use a theory which is weaker in some respects and stronger in others.

In the rest of this section, we comment briefly on the first two points. We'll develop point (c) in the final section.

(a) Since we are now going to be talking about different theories let's use subscripts to keep track: so Con_T is the canonical consistency statement for T, constructed on the lines of $\mathsf{Con}_{\mathsf{PA}}$ (our original sentence Con for PA). In other words, $\mathsf{Con}_T =_{\text{def}} \neg\mathsf{Prov}_T(\ulcorner\bot\urcorner)$, where Prov_T is a canonical provability predicate for T constructed along the lines of Prov for PA.

Here then is an immediate corollary of our key Theorem:

> **Theorem 24.1*** *If T is a consistent sub-theory of* PA *then* $T \nvdash \mathsf{Con}_{\mathsf{PA}}$.

A sub-theory of PA is any theory whose theorems are a subset of the theorems of PA.

Evidently, if the sub-theory T doesn't have enough of the language of basic arithmetic, then it won't even be able to frame the PA-wff we've abbreviated $\mathsf{Con}_{\mathsf{PA}}$; so it certainly won't be able to prove it. The more interesting case is where T is a theory which does share the language of PA but doesn't have all the induction axioms and/or uses a weaker deductive system than classical first-order logic. Such a theory T can't prove *more* than PA. So, by Theorem 24.1, assuming T is consistent, T can't prove $\mathsf{Con}_{\mathsf{PA}}$ either.

Recall our brief discussion in Section 10.8, where we first raised the issue of PA's consistency. We noted that arguing for consistency by appealing to the existence of a putative interpretation might perhaps be thought risky (we might worry that the appeal is potentially vulnerable to the discovery that our intuitions are deceptive and that there is a lurking incoherence in the interpretation). So the question naturally arises whether we can give a demonstration of PA's consistency that depends on something supposedly more secure. And once we've got the idea of coding up facts about provability using Gödel numbering, we might wonder whether we could, so to speak, lever ourselves up to establishing PA's consistency (not its truth, but at least its consistency) by assuming the truth of some weaker and supposedly less problematic arithmetic[4] and trying to prove $\mathsf{Con}_{\mathsf{PA}}$. Theorem 24.1* shows that this can't be done.

(b) Here's another corollary of our Theorem:

> **Theorem 24.1**** *If T extends* PA, *then* $\mathsf{PA} \nvdash \mathsf{Con}_T$.

That's because, if PA could establish the consistency of the stronger theory T, it would thereby establish the consistency of PA as part of that theory, contrary to the Second Theorem. (To prove Theorem 24.1** properly, we could use lemma (L) of Section 25.6.)

We'll be returning to consider the significance of this second corollary in the next Interlude. It matters crucially for the assessment of Hilbert's Programme, which we briefly mentioned in Section 1.6. But the key point is already clear: we *can't* take some problematic rich theory which extends arithmetic (set theory,

[4]Such as the minimally inductive theory $\mathsf{I}\Delta_0$ which we defined in Section 10.4, fn. 3.

for example) and show that it is consistent by (i) talking about its proofs using coding tricks and then (ii) using uncontentious reasoning already available in some relatively weak, purely arithmetical, theory.

24.7 Proving the consistency of PA

Trying to prove the consistency of PA by appeal to a *stronger* theory which already contains PA might well not seem to be a good strategy if we want to quiet doubts about PA's consistency (for doubts about PA will carry over to become doubts about the stronger theory). And the Second Theorem shows that it is impossible to prove PA's consistency by appeal to a *weaker* theory which is contained inside PA. But, as we noted, there's another possibility: maybe we can prove PA's consistency by appeal to an attractive theory which is weaker than PA in some respects but even though stronger in others.

And this is, in effect, what Gerhard Gentzen did in his consistency proof for arithmetic.[5] However, to explore his type of argument at any length would sadly take us far too far away from our main themes. So here's just a very sketchy outline, due to Gentzen himself. As with our sketched proof of Goodstein's Theorem in Section 23.2, if you don't know much about the ordinals, you'll probably get little from these few brief hints. But don't worry. It is only the overall *structure* of the argument that matters.

We start with a 'sequent proof' formulation of PA, a formulation which can easily seen to be equivalent to our official Hilbert-style version.

> The 'correctness' of a proof [in particular, the lack of contradiction] depends on the correctness of certain other simpler proofs contained in it as special cases or constituent parts. This fact motivates the arrangement of proofs in linear *order* in such a way that those proofs on whose correctness the correctness of another proof depends precede the latter proof in the sequence. This arrangement of the proofs is brought about by correlating with each proof a certain transfinite ordinal number.

But why is the relevant linear ordering of proofs *transfinite* (in other words, why must it allow an item in the ordering to have an infinite number of predecessors)? Because

> [it] may happen that the correctness of a proof depends on the correctness of infinitely many simpler proofs. An example: Suppose that in the proof a proposition is proved for *all* natural numbers by complete induction. In that case the correctness of the

[5] Gentzen's key papers (1936, 1938) are difficult: but the headline news is given wonderfully clearly in his wide-ranging lecture on 'The concept of infinity in mathematics' (Gentzen, 1937, pp. 230–233, from which we are quoting here). For later variants on Gentzen's proof, see Mendelson (1964, Appendix), or – at greater length – Pohlers (1989, Ch. 1), Takeuti (1987, Ch. 2).

> proof obviously depends on the correctness of every single one of the infinitely many individual proofs obtained by specializing to a particular natural number. Here a natural number is insufficient as an ordinal number for the proof, since each natural number is preceded by only finitely many other numbers in the natural ordering. We therefore need the transfinite ordinal numbers in order to represent the natural ordering of the proofs according to their complexity.

And now the key step is to argue by an induction along the transfinite ordering of proofs. The very simplest proofs at the beginning of the ordering transparently can't lead to contradiction. Then it is easy to see that

> once the correctness of all proofs preceding a particular proof in the sequence has been established, the proof in question is also correct precisely because the ordering was chosen in such a way that the correctness of a proof depends on the correctness of certain earlier proofs. From this we can now obviously infer the correctness of all proofs by means of a transfinite induction, and we have thus proved, in particular, the desired consistency.

More precisely, the ordering of possible proof-trees that we need to use to prove the consistency of PA turns out have the order type of the ordinals less than ε_0 (i.e. the ordinals which are sums of powers of ω). So, what Gentzen's proof needs is the assumption that transfinite induction up to ε_0 is legitimate.

Now, the procedure for listing the simpler proofs on which a more complex proof depends is such that a computer could do it using only 'for' loops. If we code up the proofs by Gödel numbers, then this sort of 'reduction' task can be handled by p.r. functions. But PA can deal with p.r. functions and deal with codings of facts about proofs: so the one place that the Gentzen-style proof invokes an idea that isn't available in PA is where it appeals to transfinite induction up to ε_0. So a minimal theory in which we can carry out the proof is one in which we can handle primitive recursive functions *and* handle enough transfinite induction, maybe via coding tricks. For example, it turns out to be enough to take the simple theory PRA_0 (that we met back in Section 12.4) and add on enough to deal with transfinite induction for quantifier-free formulae. This theory is neither contained in PA (since it can prove $\mathsf{Con}_{\mathsf{PA}}$ by Gentzen's method, which PA can't), nor does it contain PA (since it can't prove more complex, quantifier-involving, instances of the Induction Schema).

Of course, is a very moot point whether – if you were actually worried about the consistency of PA – this proof when fully spelt out would help resolve your doubts. For, if you were worried whether the use of induction in general could lead to contradiction, then appealing to an argument which deploys an induction principle can hardly help! But perhaps your worry was about our 'in for a penny, in for a pound' argument in Section 10.4 (we argued, recall, that the basic

motivation for induction should encourage us to generously allow instances of the Schema of arbitrary complexity): perhaps you thought that induction over arbitrarily complex wffs might engender trouble. In that case, the fact that we can show that PA is consistent using an induction principle which is only applied to low-complexity wffs (even though the induction runs over more than just the natural numbers) could possibly soothe your worries.

Be that as it may: the Gentzen proof is a fascinating achievement, containing the seeds of wonderful modern work in proof theory. But the full story will have to be left for another occasion.

25 The derivability conditions

In the last chapter, we gave some headline news about the Second Theorem for PA, and about how it rests on the Formalized First Theorem. In this action-packed chapter, we'll say something about how the Formalized First Theorem is proved. More exactly, we state the so-called Hilbert-Bernays-Löb derivability conditions on the provability predicate for theory T and show that these suffice for proving the Formalized First Theorem for T, and hence the Second Theorem. We'll also prove a very nice theorem due to Löb.

25.1 More notation

To improve readability, let's introduce some neat notation. We will henceforth abbreviate $\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$ simply by $\Box_T\varphi$.[1] So Con_T can now alternatively be abbreviated as $\neg\Box_T\bot$.

Two comments. First, note that our box symbol actually does a double job: it further abbreviates the long predicative expression already abbreviated by Prov_T, *and* it absorbs the corner quotes that turn a wff into the standard numeral for that wff's Gödel number. If you are logically pernickety, then you might be rather upset about introducing a notation which in this way rather disguises the complex logical character of what is going on.[2] But my line is that abbreviatory convenience here trumps notational perfectionism.

Second, we will very often drop the explicit subscript from the box symbol, and let context supply it. We'll also drop other subscripts in obvious ways. For example, here's the Formalized First Theorem for theory T in our new notation:

$$T \vdash \mathsf{Con} \rightarrow \neg\Box\mathsf{G}.$$

The turnstile signifies the existence of a proof in T's deductive system. And the box in this statement is then to be understood by default as expressing

[1] If you are familiar with modal logic, then you will immediately recognize the conventional symbol for the necessity operator. And the parallels and differences between '"$1 + 1 = 2$" is provable (in T)' and 'It is necessarily true that $1 + 1 = 2$' are highly suggestive. These parallels and differences are the topic of 'provability logic', the subject of a contemporary classic (Boolos, 1993).

[2] The beef is this. The notation '$\Box\varphi$' *looks* as if it ought to be a complex wff embedding φ, so that as φ increases in logical complexity, so does $\Box\varphi$. But not so. However complex φ is, $\ulcorner\varphi\urcorner$ is just a numeral and $\Box\varphi$, i.e. $\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$, stays resolutely a Σ_1 wff.

Perhaps the logically kosher approach is not to regard the box as an abbreviation, but to introduce a *new* modal language, and then explore a mapping relation that links modal sentences to arithmetical 'realizations' via a $\Box$/Prov link. For a properly careful treatment of this, see Boolos (1993) again.

provability in T, while Con and G are (of course) respectively the canonical consistency sentence Con_T and the Gödel sentence G_T for that same theory.

25.2 The Hilbert-Bernays-Löb derivability conditions

(a) We haven't yet *proved* the Formalized First Theorem for PA. And, as we remarked before, Gödel himself didn't prove the corresponding result for his particular formal theory P. The hard work was first done by David Hilbert and Paul Bernays in their *Grundlagen der Mathematik* (1939): the details of their proof are in fact due to Bernays, who had discussed it with Gödel during a transatlantic voyage.

Hilbert and Bernays helpfully isolated three conditions on the predicate Prov_T, conditions whose satisfaction is enough for a nice theory T to prove the Formalized First Theorem. Later, Martin H. Löb (1955) gave a rather neater version of these conditions. Here they are in Löb's version, and in our snappy notation:

C1. If $T \vdash \varphi$, then $T \vdash \Box\varphi$,

C2. $T \vdash \Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$,

C3. $T \vdash \Box\varphi \to \Box\Box\varphi$,

where φ and ψ are, of course, any sentences. Henceforth, we'll call these the *derivability conditions*. We'll see in the next chapter that they hold for PA.

(b) Why are these a very natural set of conditions to propose? Well, think about our (re)proof of the first part of the First Theorem in Section 21.1. We started from the special case of the Diagonalization Lemma

D. $T \vdash \mathsf{G} \leftrightarrow \neg\Box\mathsf{G}$.

Or rather, we started from one half that result,

D′. $T \vdash \mathsf{G} \to \neg\Box\mathsf{G}$,

and we put that together with the principle (C1) in order to derive $T \nvdash \mathsf{G}_T$.

Now, given that (D′) and (C1) imply half of the First Theorem, we might reasonably expect to be able to argue from the claims that T *'knows' that (D′) holds* and T *'knows' that (C1) holds* to the conclusion T *'knows' that half the First Theorem holds*.

Let's put that more carefully! The thought that T 'knows' that (D′) holds is tantamount to

K. $T \vdash \Box(\mathsf{G} \to \neg\Box\mathsf{G})$,

And that follows from (D′) by (C1). The thought that T 'knows' that (C1) holds is captured by (C3). The hoped for inference is therefore from (K) and (C3) to the Formalized First Theorem. That should go through so long as T is able

to cope with the idea that if $\varphi \to \psi$ and φ are both provable, then so is their consequence ψ. Which is just what (C2) requires.

So, putting everything together, the three derivability conditions – together with the special case of the Diagonalization Lemma which is provable for any nice theory – look as if they ought to give us the Formalized First Theorem.

(c) And indeed they do, as we'll now prove.

Theorem 25.1 *If T is nice and the derivability conditions hold, then $T \vdash \mathsf{Con} \to \neg\Box\mathsf{G}$.*

Proof First, if T is nice, (D) is true by Theorem 20.3. And second, note that for any nice theory T, $T \vdash \neg\bot$ (either by the built-in logic, or because we've put $\bot =_{\text{def}} \mathsf{0} = \mathsf{1}$). So simple logic shows that, for any wff φ, we have

$$T \vdash \neg\varphi \to (\varphi \to \bot).$$

Given the latter and (C1), this means

$$T \vdash \Box(\neg\varphi \to (\varphi \to \bot)).$$

So given (C2) and using modus ponens, it follows that for any φ

A. $T \vdash \Box\neg\varphi \to \Box(\varphi \to \bot)$.

We now argue as follows:

1.	$T \vdash \mathsf{G} \to \neg\Box\mathsf{G}$	Half of D
2.	$T \vdash \Box(\mathsf{G} \to \neg\Box\mathsf{G})$	From 1, given C1
3.	$T \vdash \Box\mathsf{G} \to \Box\neg\Box\mathsf{G}$	From 2, given C2
4.	$T \vdash \Box\neg\Box\mathsf{G} \to \Box(\Box\mathsf{G} \to \bot)$	Instance of A
5.	$T \vdash \Box\mathsf{G} \to \Box(\Box\mathsf{G} \to \bot)$	From 3 and 4
6.	$T \vdash \Box\mathsf{G} \to (\Box\Box\mathsf{G} \to \Box\bot)$	From 5, given C2
7.	$T \vdash \Box\mathsf{G} \to \Box\Box\mathsf{G}$	Instance of C3
8.	$T \vdash \Box\mathsf{G} \to \Box\bot$	From 6 and 7
9.	$T \vdash \neg\Box\bot \to \neg\Box\mathsf{G}$	Contraposing
10.	$T \vdash \mathsf{Con} \to \neg\Box\mathsf{G}$	Definition of Con

Which gives us our general version of the Formalized First Theorem. ⊠

(d) We can now immediately derive a version of the Second Theorem for T:

Theorem 25.2 *If T is nice and the derivability conditions hold, then $T \nvdash \mathsf{Con}_T$.*

Proof The argument goes as before:

1.	$T \vdash \mathsf{Con} \to \neg\Box\mathsf{G}$	Just proved
2.	$T \vdash \neg\Box\mathsf{G} \to \mathsf{G}$	Other half of D
3.	$T \vdash \mathsf{Con} \to \mathsf{G}$	From 1 and 2
4.	$T \nvdash \mathsf{G}$	The First Theorem!
5.	$T \nvdash \mathsf{Con}$	From 3 and 4 ⊠

(e) Before proceeding, let's also note another simple result:

> **Theorem 25.3** *If T is nice and the derivability conditions hold, then for any sentence φ, $T \vdash \neg\Box\varphi \rightarrow \mathsf{Con}$.*

So T knows that, *if* it can't prove φ, it must be consistent.

Proof We argue as follows:[3]

1.	$T \vdash \bot \rightarrow \varphi$	Logic!
2.	$T \vdash \Box(\bot \rightarrow \varphi)$	From 1, given C1
3.	$T \vdash \Box\bot \rightarrow \Box\varphi$	From 2, given C2
4.	$T \vdash \neg\Box\varphi \rightarrow \neg\Box\bot$	Contraposing
5.	$T \vdash \neg\Box\varphi \rightarrow \mathsf{Con}$	Definition of Con ☒

So, since T can't prove Con, T doesn't entail $\neg\Box\varphi$ for any φ at all. Hence T doesn't ever 'know' that it can't prove φ, even when it can't.

In sum, suppose that T is nice and the derivability conditions hold: by (C1), T knows all about what it *can* prove; but we've now shown that it knows nothing about what it *can't* prove.

25.3 G, Con, and 'Gödel sentences'

Suppose T is nice and the derivability conditions hold. Then Theorem 25.1 tells us that $T \vdash \mathsf{Con} \rightarrow \neg\Box\mathsf{G}$; and, from Theorem 25.3, it immediately follows that $T \vdash \neg\Box\,\mathsf{G} \rightarrow \mathsf{Con}$. Put those two results together with (D), the special case of the Diagonalization Lemma, and we get:

> **Theorem 25.4** *If T is nice and the derivability conditions hold, then $T \vdash \mathsf{Con} \leftrightarrow \mathsf{G}$.*

Now consider the following argument:

1.	$T \vdash \mathsf{Con} \rightarrow \mathsf{G}$	Just shown
2.	$T \vdash \Box(\mathsf{Con} \rightarrow \mathsf{G})$	From 1, by C1
3.	$T \vdash \Box\mathsf{Con} \rightarrow \Box\mathsf{G}$	From 2, by C2
4.	$T \vdash \mathsf{Con} \rightarrow \neg\Box\mathsf{G}$	Thm. 25.1 again
5.	$T \vdash \mathsf{Con} \rightarrow \neg\Box\mathsf{Con}$	From 3 and 4

But Theorem 25.3 tells us that $T \vdash \neg\Box\mathsf{Con} \rightarrow \mathsf{Con}$. Which establishes

> **Theorem 25.5** *If T is nice and the derivability conditions hold, then $T \vdash \mathsf{Con} \leftrightarrow \neg\Box\mathsf{Con}$.*

In other symbols, $T \vdash \mathsf{Con} \leftrightarrow \neg\mathsf{Prov}(\ulcorner\mathsf{Con}\urcorner)$. Hence, on the same assumptions about T, Con *is also a 'fixed point' for* $\neg\mathsf{Prov}(\mathsf{x})$ in the sense of Section 20.4.

[3] In fact, the only aspect of T's 'niceness' that we use is that it contains enough logic.

Two comments. First, suppose we use the phrase 'Gödel sentence' in a wide sense to refer to any fixed point for $\neg\mathsf{Prov}_T$ (see Section 21.2). Then we've just shown that there are Gödel sentences like Con which are *not* self-referential in any way, however loosely interpreted. But, as we noted in Section 21.1, *any* fixed point for $\neg\mathsf{Prov}_T$ will be formally undecidable (assuming T is ω-consistent). *So there are formally undecidable sentences which aren't self-referential.* That observation should scotch once and for all any lingering suspicion that the incompleteness phenomena are somehow inevitably tainted by self-referential paradox.

Second, note that our demonstration that $T \vdash \mathsf{Con} \rightarrow \neg\Box\mathsf{Con}$ tells us that T can itself prove a wff that expresses the Second Theorem, i.e. a wff that says 'If T is consistent, then it cannot prove Con'. We can call this result the *Formalized Second Theorem.*

And that's the main business of this chapter done. However, it's worth adding four more sections on related matters. The next section uses Theorem 25.4 to make an observation about sequences of ever-richer but still incomplete theories. The following section says more about fixed points for $\neg\mathsf{Prov}$. The one after that is about theories that 'prove' their own inconsistency. And the final section of this chapter presents Löb's elegant Theorem. But you can skip them all on a first reading. Don't get bogged down in these additional details!

25.4 Incompletability and consistency extensions

As we've noted before, if we start from PA, and add its canonical Gödel sentence G as a new axiom, the resulting theory $\mathsf{PA} + \mathsf{G}$ trivially proves PA's Gödel sentence. But the new theory has its own canonical Gödel sentence $\mathsf{G}_{\mathsf{PA}+\mathsf{G}}$, and so must remain incomplete. We can then, of course, construct a yet richer theory $\mathsf{PA} + \mathsf{G} + \mathsf{G}_{\mathsf{PA}+\mathsf{G}}$, which can prove the Gödel sentences of both the two preceding theories: but this theory too has its own Gödel sentence and is incomplete. And so it goes. We can continue in the same vein, forming an infinite sequence of theories, each theory trivially proving the canonical Gödel sentences of its predecessors but remaining incomplete and incompletable.

Now, we've just shown that, if T is nice and the derivability conditions hold, Con_T and G_T are provably equivalent in T. And we'll later be showing that the derivability conditions indeed hold for theories extending PA. So our sequence of theories got by adding Gödel sentences is equivalent to the sequence PA, $\mathsf{PA} + \mathsf{Con}$, $\mathsf{PA} + \mathsf{Con} + \mathsf{Con}_{\mathsf{PA}+\mathsf{Con}}$, ..., where the $n+1$-th theory on the list adds the canonical consistency sentence of the n-th theory as a new axiom. So, each theory in the sequence proves the canonical consistency sentences of its predecessors, but still remains incomplete and incompletable.

The interest in putting things in terms of 'consistency extensions' rather than in terms of the addition of Gödel sentences lies in the following thought. Suppose we accept that PA is sound. Then, reflecting on this, we'll readily come to accept that $\mathsf{PA} + \mathsf{Con}$ is sound too – and we don't need to go through any intricate

Gödelian arguments to get to *that* realization (the argument is elementary although it does involve us in more than purely arithmetical reasoning because we have to think about our own theorizing). And reflecting on that, we'll readily come to accept that $\mathsf{PA} + \mathsf{Con} + \mathsf{Con}_{\mathsf{PA}+\mathsf{Con}}$ is sound too. And so on. So starting from our initial thought that PA is sound, reflection can very naturally drive us along the sequence of consistency extensions. But still the theories we arrive at by pursuing this natural line of thought remain incomplete and incompletable.

And what if we consider 'consistency extensions' of PA which add not just a finite number of such consistency statements, but all of them together? What if we now add the consistency statement for that infinitely augmented theory, and keep on going again after that? No matter. So long as our theory remains p.r. axiomatized, it must remain incomplete.[4]

25.5 The equivalence of fixed points for ¬Prov

G and Con are both fixed points for $\neg\mathsf{Prov}(\mathsf{x})$. They are also provably equivalent. We might wonder: are *all* the fixed points for $\neg\mathsf{Prov}(\mathsf{x})$ provably equivalent? The answer is given by

> **Theorem 25.6** *Given that the derivability conditions hold for T, all the fixed points for $\neg\mathsf{Prov}_T(\mathsf{x})$ are provably equivalent in T to Con_T.*

Proof We assume $T \vdash \gamma \leftrightarrow \neg\Box\gamma$ and need to derive $T \vdash \gamma \leftrightarrow \neg\Box\bot$. The derivation is straightforward: try it as a brainteaser before reading on!

1.	$T \vdash \gamma \leftrightarrow \neg\Box\gamma$	Premiss
2.	$T \vdash \gamma \rightarrow (\Box\gamma \rightarrow \bot)$	From 1, by logic
3.	$T \vdash \Box(\gamma \rightarrow (\Box\gamma \rightarrow \bot))$	From 2, given C1
4.	$T \vdash \Box\gamma \rightarrow (\Box\Box\gamma \rightarrow \Box\bot)$	From 3, given C2
5.	$T \vdash \Box\gamma \rightarrow \Box\Box\gamma$	From C3
6.	$T \vdash \Box\gamma \rightarrow \Box\bot$	From 4 and 5
7.	$T \vdash \bot \rightarrow \gamma$	Logic
8.	$T \vdash \Box(\bot \rightarrow \gamma)$	From 7, given C1
9.	$T \vdash \Box\bot \rightarrow \Box\gamma$	From 8, given C2
10.	$T \vdash \Box\gamma \leftrightarrow \Box\bot$	From 6 and 9
11.	$T \vdash \gamma \leftrightarrow \neg\Box\bot$	From 1 and 10 ⊠

One quick comment to link this little result to an important observation we made in Section 21.2. You might momentarily be tempted to think: 'Assuming T is nice, its canonical consistency sentence Con_T is true, as is its canonical

[4] Which is not at all to say that the general theory of consistency extensions isn't interesting, for there are troublesome complications, classically explored in Feferman (1960). For a recent discussion, see Franzén (2004).

Gödel sentence G_T. But we've just seen that all of T's other Gödel sentences – in the sense of all the fixed points for $\neg\mathsf{Prov}(\mathsf{x})_T$ – are provably equivalent to these truths. So these other Gödel sentences must be true too.' But not so: that forgets that a nice T may be a consistent-but-*unsound* theory. Such a T will have some false Gödel sentences mixed in with the true ones (and hence some false biconditional theorems relating the false ones to the true ones).

25.6 Theories that 'prove' their own inconsistency

An ω-consistent T can't prove $\neg\mathsf{Con}_T$ (that was Theorem 24.2). By contrast, a consistent but ω-*inconsistent* T might well have $\neg\mathsf{Con}_T$ as a theorem.

The proof is pretty trivial, once we note a simple lemma. Suppose S and R are two p.r. axiomatized theories, which share a deductive logic; and suppose every axiom of the simpler theory S is also an axiom of the richer theory R. Evidently, if the richer R is consistent, then the simpler S must be consistent too. And the arithmetical claim that encodes this fact can be formally proved. Contraposing,

L. $\mathsf{PA} \vdash \neg\mathsf{Con}_S \to \neg\mathsf{Con}_R$.

Proof sketch Suppose $\neg\mathsf{Con}_S$, i.e. suppose $\exists\mathsf{v}\,\mathsf{Prf}_S(\mathsf{v}, \bot)$. Hence for some a, $\mathsf{Prf}_S(\mathsf{a}, \bot)$. And that implies $\mathsf{Prf}_R(\mathsf{a}, \bot)$. Why? Because the difference between the unpacked definitions of Prf_S and Prf_R – the definitions which formally reflect what counts as (the code for) an S proof and an R proof – will just be that the latter needs some more disjuncts to allow for the extra axioms that can be invoked in an R proof. So it follows that $\exists\mathsf{v}\,\mathsf{Prf}_R(\mathsf{v}, \bot)$, i.e. $\neg\mathsf{Con}_R$. And the inferences here only uses the first-order logic built into PA. ⊠

Now let's put lemma (L) to use. Take the simpler theory S to be PA and let the richer theory R be PA augmented by the extra axiom $\neg\mathsf{Con}_{\mathsf{PA}}$. By definition, $R \vdash \neg\mathsf{Con}_{\mathsf{PA}}$. So using our lemma we can conclude $R \vdash \neg\mathsf{Con}_R$. R is ω-inconsistent (why? cf. Theorem 24.2). But it is consistent if PA is (why? because we know from the Second Theorem that if R proved a contradiction, and hence $\mathsf{PA} \vdash \mathsf{Con}_{\mathsf{PA}}$, then PA would be inconsistent). So, assuming PA is consistent,

1. R is a consistent theory which 'proves' its own inconsistency.

And since R proves $\neg\mathsf{Con}_R$,

2. there is a *consistent* theory R such that $R + \mathsf{Con}_R$ is *inconsistent*.

What are we to make of these apparent absurdities? Well, giving the language of R its standard arithmetical interpretation, the theory is just wrong in what it says about its inconsistency! But on reflection that shouldn't be much of a surprise. Believing, as we no doubt do, that PA is consistent, we already know that the theory R gets things wrong right at the outset, since it has the false axiom $\neg\mathsf{Con}_{\mathsf{PA}}$. So R doesn't really *prove* (establish-as-true) its own inconsistency, since we don't accept the theory as correct on the standard interpretation.

By the way, we will see in the next chapter that the derivability conditions hold for theories that contain PA, so they will hold for R. Hence by Theorem 25.4, $R \vdash \mathsf{Con}_R \leftrightarrow \mathsf{G}_R$. So since $R \vdash \neg\mathsf{Con}_R$, $R \vdash \neg\mathsf{G}_R$. Hence the ω-inconsistent R also 'disproves' its own true canonical Gödel sentence. That's why the requirement of ω-inconsistency – or at least 1-consistency – *has* to be assumed in the proof that arithmetic is incomplete, if we are prove it by constructing an original-style Gödel sentence like G_R.

25.7 Löb's Theorem

(a) Finally in this busy chapter, let's think about the following issue. Suppose T is a nice *sound* theory. Then if φ is a provable sentence, φ is true. So each instance of the so-called *reflection schema* for T, i.e. each instance of

$$\Box_T\varphi \to \varphi,$$

will be true. Hence we would *like* T to be able to prove *all* such instances – since we'd like a nice sound theory T to prove as many arithmetic truths as possible. The question is: can it?

It is easy to see that the answer must be 'no':

> **Theorem 25.7** *If T is nice, then T cannot prove all instances of the reflection schema for T.*

Proof Suppose $T \vdash \Box\mathsf{G} \to \mathsf{G}$ (for a canonical Gödel sentence G). Theorem 20.3 tells us that if T is nice, then it is also true that $T \vdash \mathsf{G} \leftrightarrow \neg\Box\mathsf{G}$. So, given T is nice, $T \vdash \Box\mathsf{G} \to \neg\Box\mathsf{G}$, hence $T \vdash \neg\Box\mathsf{G}$, hence $T \vdash \mathsf{G}$, contradicting the First Theorem. Therefore, if T is nice, $T \nvdash \Box\mathsf{G} \to \mathsf{G}$. ⊠

Two corollaries of this result are worth mentioning for future use. (1) Since G is Π_1, we've shown that T can't prove even just all instances of the reflection schema where φ is Π_1. And (2), if T' is a richer theory that extends T, then T certainly can't prove all instances of the reflection schema for the richer T', i.e. all instances of $\Box_{T'}\varphi \to \varphi$, even just those where φ is Π_1.

But have we just noted a rather special blind spot for theories? Can they otherwise prove all the instances of their own reflection principle for non-Gödelian φ? This question is answered, in the negative again, by *Löb's Theorem:*

> **Theorem 25.8** *If T is nice and the derivability conditions hold, then if $T \vdash \Box\varphi \to \varphi$ then $T \vdash \varphi$.*

So for *any* φ that T can't prove, T can't prove $\Box\varphi \to \varphi$ either.

(b) Löb's theorem also settles another question, one asked by Henkin (1952). By the Diagonalization Lemma applied to the unnegated wff $\mathsf{Prov}(\mathsf{z})$, there is a sentence H such that $T \vdash \mathsf{H} \leftrightarrow \mathsf{Prov}(\ulcorner\mathsf{H}\urcorner)$ – i.e., we can use diagonalization again

to construct such a sentence H that 'says' that it is provable.[5] Henkin asked: *is* H provable?

It is. For by hypothesis, $T \vdash \mathsf{Prov}(\ulcorner \mathsf{H} \urcorner) \rightarrow \mathsf{H}$, i.e. $T \vdash \Box\mathsf{H} \rightarrow \mathsf{H}$; so $T \vdash \mathsf{H}$ by Löb's Theorem.

(c) We can easily derive Löb's Theorem as a consequence of the Second Incompleteness Theorem as follows (the argument is due to Kreisel):

Proof sketch Assume we are dealing with arithmetics to which the Second Theorem applies. We'll suppose $T \vdash \Box_T\varphi \rightarrow \varphi$, and derive $T \vdash \varphi$.

So make that supposition and now consider the theory T' you get by taking T and adding $\neg\varphi$ as a new axiom. Then trivially $T' \vdash \neg\varphi$ and $T' \vdash \Box_T\varphi \rightarrow \varphi$, so $T' \vdash \neg\Box_T\varphi$. But to prove that φ is unprovable in T is to prove that adding $\neg\varphi$ to T doesn't lead to contradiction, i.e. is to prove that T' is consistent. So for T' to prove $\neg\Box_T\varphi$ is tantamount to proving its own consistency. But by the Second Incompleteness Theorem, T' can't prove its own consistency if it is itself consistent. So T' is inconsistent. Maybe T is already inconsistent. Else adding $\neg\varphi$ to T leads to inconsistency. So either way $T \vdash \varphi$. ⊠

(d) However, it is a lot more fun to proceed the other way around; so now we are going to use the derivability conditions to prove Löb's Theorem directly. We will then re-establish the Second Incompleteness Theorem as a corollary.

Proof Assume that, for a given φ,

1. $T \vdash \Box\varphi \rightarrow \varphi$.

Now consider the wff $(\mathsf{Prov}(\mathsf{z}) \rightarrow \varphi)$. By hypothesis, T is nice, so we can invoke the general Diagonalization Lemma and apply it to this wff. Hence for some γ, T proves $\gamma \leftrightarrow (\mathsf{Prov}(\ulcorner\gamma\urcorner) \rightarrow \varphi)$. Or, in our new notation,

2.	$T \vdash \gamma \leftrightarrow (\Box\gamma \rightarrow \varphi)$	
3.	$T \vdash \gamma \rightarrow (\Box\gamma \rightarrow \varphi)$	From 2
4.	$T \vdash \Box(\gamma \rightarrow (\Box\gamma \rightarrow \varphi))$	From 3, by C1
5.	$T \vdash \Box\gamma \rightarrow \Box(\Box\gamma \rightarrow \varphi)$	From 4, by C2
6.	$T \vdash \Box\gamma \rightarrow (\Box\Box\gamma \rightarrow \Box\varphi)$	From 5, by C2
7.	$T \vdash \Box\gamma \rightarrow \Box\Box\gamma$	By C3
8.	$T \vdash \Box\gamma \rightarrow \Box\varphi$	From 6 and 7
9.	$T \vdash \Box\gamma \rightarrow \varphi$	From 1 and 8
10.	$T \vdash \gamma$	From 2 and 9
11.	$T \vdash \Box\gamma$	From 10, by C1
12.	$T \vdash \varphi$	From 9 and 11

Hence, if $T \vdash \Box\varphi \rightarrow \varphi$, then $T \vdash \varphi$. But φ was an arbitrarily selected wff; so we are done. ⊠

[5] If the Gödel sentence G is reminiscent of the Liar sentence 'This sentence is false', then Henkin's sentence H is reminiscent of the Truth-teller sentence 'This sentence is true'. For discussion of the Truth-teller, see e.g. Simmons (1993) or Yaqub (1993).

And as we said, we can now get from Löb's Theorem back to the Second Incompleteness Theorem. The argument is swift:

Proof If $T \vdash \mathsf{Con}$ then, as trivial consequence, $T \vdash \neg\mathsf{Con} \rightarrow \bot$, i.e. $T \vdash \Box\bot \rightarrow \bot$ Then, by Löb's Theorem, we can conclude $T \vdash \bot$. Hence, if $T \vdash \mathsf{Con}$, T is inconsistent. ⊠

(e) Now, we noted at the very outset that the reasoning for Gödel's First Theorem has echoes of the Liar paradox. Let's finish this chapter by noting that the proof of Löb's theorem (which we've just seen is tantamount to the Second Theorem) echoes another logical paradox.

Suppose first we temporarily reinterpret the symbol '$\Box$' as expressing the truth-predicate, so we read $\Box S$ as *'S' is true*. Second, let φ express any proposition you like, e.g. *The moon is made of green cheese*. Then

$1'$. $\quad \Box\varphi \rightarrow \varphi$

is a truism about truth. Third, suppose that the sentence γ says: *if γ is true, then the moon is made of green cheese*. Then we have by definition

$2'$. $\quad \gamma \leftrightarrow (\Box\gamma \rightarrow \varphi)$.

From here on, we can argue as before (deleting the initial '$T \vdash$'), appealing to conditions (C1) to (C3) now reinterpreted as intuitive principles about truth:

C1′. if φ then φ is true;

C2′. if $\varphi \rightarrow \psi$ is true, then if φ is true, ψ is true too;

C3′. if φ is true, then it is true that φ is true.

Using these evidently sound principles we again arrive by exactly the same route at

$12'$. $\quad \varphi$.

So, from truisms about truth and a definitional equivalence like $(2')$ we can, it seems, prove that the moon is made of green cheese. Or prove anything else you like, since the interpretation of φ was arbitrary. (Exercise: check through the details.)

This line of reasoning is nowadays usually known as 'Curry's Paradox' – after Haskell B. Curry who presented it in Curry (1942) – though close relations of it were certainly known to medieval logicians such as Albert of Saxony.[6] It isn't obvious what the best way is to block Curry's paradox, any more than it is obvious what the best way is to block the Liar. There is no doubt *something* fishy about postulating a sentence γ such that $(2')$ holds: but *what* exactly?

Fortunately answering that last question is not our business. We merely remark that Löb's Theorem, like Gödel's, is not a paradox but a limitative result, a result about a theory's inability to prove certain propositions about its own provability properties.

[6] See his terrific *Insolubilia* of 1490, translated in Kretzmann and Stump (1988).

26 Deriving the derivability conditions

We have seen that the Second Theorem obtains for any nice theory T which also satisfies the Hilbert-Bernays-Löb derivability conditions

C1. if $T \vdash \varphi$, then $T \vdash \Box\varphi$,

C2. $T \vdash \Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$,

C3. $T \vdash \Box\varphi \to \Box\Box\varphi$,

where φ, ψ are any T-sentences. But we already know from Theorem 20.1 that (C1) holds for any nice theory: indeed, if $T \vdash \varphi$, even $\mathsf{Q} \vdash \Box_T\varphi$. So the obvious next question is: what does it take for conditions (C2) and (C3) to obtain as well?

26.1 Nice* theories

$\Box\varphi$ – i.e. $\Box_T\varphi$ – abbreviates $\exists \mathsf{v}\, \mathsf{Prf}_T(\mathsf{v}, \ulcorner\varphi\urcorner)$. By definition, Prf_T canonically captures the relation Prf_T. So statements involving $\Box_T$ make claims about the Prf_T relation in a very direct way. And it looks quite plausible that T will be able to prove enough facts about Prf_T to make (C2) and (C3) true *if T is sufficently rich*, and in particular if it has sufficient inductive strength.

Whatever the details, however, in order to prove (C2) and (C3) we are surely not going to need to invoke anything like the full range of induction axioms that are available in e.g. PA. For Prf_T is a Σ_1 predicate, that is built up using other Σ_1 predicates which capture the simpler functions involved in Prf_T's p.r. definition. It therefore seems quite a good bet that *the derivability conditions for T will hold even if T only has Σ_1-induction* – meaning that T's axioms include (the universal closures of) all instances of the first-order Induction Schema where the induction predicate φ is Σ_1.

If T is consistent, p.r. axiomatized, and contains Q, we said it is 'nice'. Let's now say:

> A *nice** theory T is one which is nice and also includes induction at least for Σ_1 wffs.

Let $\mathsf{I}\Sigma_1$ be the theory you get by augmenting Q with Σ_1-induction. Then, equivalently,

> A *nice** theory T is a consistent, p.r. axiomatized, theory which extends $\mathsf{I}\Sigma_1$.

So, given what we've just said, the rather natural conjecture is that we can establish the following:

> **Theorem 26.1** *The Hilbert-Bernays-Löb derivability conditions hold for any nice* theory.*

But Theorem 25.2 tells us that if T is nice and satisfies the derivability conditions then $T \nvdash \mathsf{Con}_T$. Hence, if we *can* establish our conjectured new theorem, we can immediately derive an attractively neat generalized version of the Second Incompleteness Theorem:

> **Theorem 26.2** *If T is nice*, then $T \nvdash \mathsf{Con}_T$.*

26.2 The second derivability condition

To establish Theorem 26.1, we first need to see – at least in outline – how to confirm that if T is nice* then (C2) holds. In fact, we'll outline a proof of something rather stronger. So long as T has a standard logic, we have:

> **Theorem 26.3** $\mathsf{I}\Sigma_1 \vdash \Box_T(\varphi \to \psi) \to (\Box_T\varphi \to \Box_T\psi)$.

Since a nice* T proves everything $\mathsf{I}\Sigma_1$ proves, (C2) follows.

Sketch of a proof sketch Assume that T has a Hilbert-style linear proof system (for other kinds of proof structure the following argument just gets more complicated in fiddly ways). And start with the following observation:

MP. If $Prf_T(m, \ulcorner(\varphi \to \psi)\urcorner)$ and $Prf_T(n, \ulcorner\varphi\urcorner)$, then $Prf_T(m \star n \star 2^{\ulcorner\psi\urcorner}, \ulcorner\psi\urcorner)$.

That holds because, if m numbers a T-proof of the sentence $(\varphi \to \psi)$, and n numbers a T-proof of φ, then $m \star n \star 2^{\ulcorner\psi\urcorner}$ numbers the sequence of wffs you get by writing down the proof of $(\varphi \to \psi)$ followed by the proof of φ followed by the one-wff sequence ψ. But this longer sequence is, of course, a T-proof of the sentence ψ by modus ponens.

What we now need to show is that this line of reasoning can be reflected inside $\mathsf{I}\Sigma_1$. In other words,

MP′. We can argue inside $\mathsf{I}\Sigma_1$ from premisses of the form $\mathsf{Prf}_T(\mathsf{a}, \ulcorner(\varphi \to \psi)\urcorner)$ and $\mathsf{Prf}_T(\mathsf{b}, \ulcorner\varphi\urcorner)$ to the corresponding conclusion $\mathsf{Prf}_T(\mathsf{a} * \mathsf{b} * 2^{\ulcorner\psi\urcorner}, \ulcorner\psi\urcorner)$,

where the star sign holds the place for a formal treatment of the p.r. concatenation star function (see Section 21.3, fn. 4). Given (MP′), our theorem is then immediate.

Why? Because – dropping subscript Ts for readability – our target theorem unpacks as the following claim: we can argue inside $\mathsf{I}\Sigma_1$ from premisses of the form $\exists\mathsf{v}\,\mathsf{Prf}(\mathsf{v}, \ulcorner(\varphi \to \psi)\urcorner)$ and $\exists\mathsf{v}\,\mathsf{Prf}(\mathsf{v}, \ulcorner\varphi\urcorner)$ to the corresponding conclusion $\exists\mathsf{v}\,\mathsf{Prf}(\mathsf{v}, \ulcorner\psi\urcorner)$. And to establish that *that* holds, it is enough to argue inside

$\mathsf{I}\Sigma_1$ by (the Hilbert-style equivalent of) instantiating those two existential premisses to get $\mathsf{Prf}(\mathsf{a}, \ulcorner(\varphi \to \psi)\urcorner)$ and $\mathsf{Prf}(\mathsf{b}, \ulcorner\varphi\urcorner)$, using (MP′), then existentially generalizing the result.

So how do we show that (MP′) is actually true? Well, remember again that Prf_T *canonically* captures the p.r. relation Prf_T. This means that Prf_T is ultimately built up from component wffs which reflect the components of the p.r. definition of Prf_T. But a p.r. definition of Prf_T will have a clause that covers inferences by modus ponens (see Section 15.9 for our definition of the relation *Prf* for PA). Hence our canonical Prf_T will likewise 'know about' modus ponens proofs. And we can now exploit this fact: basically, we can extract component wffs buried inside $\mathsf{Prf}(\mathsf{a}, \ulcorner(\varphi \to \psi)\urcorner)$ and $\mathsf{Prf}(\mathsf{b}, \ulcorner\varphi\urcorner)$, and go on to deduce the various wffs which can be recombined to give us $\mathsf{Prf}(\mathsf{a} * \mathsf{b} * 2^{\ulcorner\psi\urcorner}, \ulcorner\psi\urcorner)$.

Now, in order to be able to do all this, we will have to be working in a formal theory which can prove the requisite properties of the Σ_1 wff that captures the concatenation star function, and also proves the requisite properties of the Σ_1 wff that captures the decoding function *exp*, since both wffs are involved in defining Prf. Σ_1-induction will (more than) suffice to establish those properties. So $\mathsf{I}\Sigma_1$ is enough to prove (MP′) and hence we get our theorem as claimed.

And let's not get bogged down in the further details![1] ⊠

26.3 The third derivability condition

To complete the proof of Theorem 26.1, we now need to confirm – though in even rougher outline – that if T is nice* then (C3) holds. And again, we'll gesture towards a proof of a more general result. So suppose T extends Q: then,

Theorem 26.4 *For any* Σ_1 *sentence* ψ, $\mathsf{I}\Sigma_1 \vdash \psi \to \Box_T\psi$.

But $\Box\varphi_T$ is, of course, a Σ_1 sentence whatever φ is (see Section 25.1, fn. 2). So if we put $\Box_T\varphi$ for ψ in our theorem, we get

For any φ, $\mathsf{I}\Sigma_1 \vdash \Box_T\varphi \to \Box_T\Box_T\varphi$.

Remembering that a nice* theory includes $\mathsf{I}\Sigma_1$, we then get what we want:

If T is nice* then, for any sentence φ, $T \vdash \Box\varphi \to \Box\Box\varphi$.

[1] There's no way of making the proof pretty – especially given the twists in our official story in Section 13.8 about how to construct a canonical *Prf* – which is the main reason I draw the line at spelling things out any further.

Masochists who want to complete the story for themselves can start by looking at Grandy (1977, p. 75) to get an idea of the sorts of core moves we need. Be warned, however, Grandy's construction of Bew – his version of our Prf – is simpler in its details. See also Rautenberg (2006, Sec. 7.1).

Let's emphasize: filling in the details *does* require a bit of work, as e.g. Smoryński (1977, p. 839) also emphasizes and as Grandy's and Rautenberg's treatments reveal. The very brief gestures towards proofs in e.g. Boolos (1993, p. 54) and Takeuti (1987, p. 86) are therefore slightly misleading.

Sketch of a proof sketch Recall from Section 9.7 that we showed that any theory T which includes Q is Σ_1-complete: if φ is a Σ_1 sentence which is true on the standard interpretation, then $T \vdash \varphi$. So what our new theorem says is essentially that $\mathsf{I}\Sigma_1$ 'knows' that T is Σ_1-complete. In other words, when ψ is Σ_1, $\mathsf{I}\Sigma_1$ 'knows' that $\psi \to \Box_T\psi$.

And that gives us the key to proving Theorem 26.4. We just need to show that the argument for T's Σ_1-completeness can be formalized inside $\mathsf{I}\Sigma_1$. Which, needless to say, is a great deal easier said than done!

The original argument for Q's Σ_1-completeness works by an informal induction on the complexity of the relevant wffs. In other words, we first showed that Q can prove the simplest true Σ_1-wffs; and then we showed that Q can prove more complex Σ_1 truths built up using the connectives and bounded quantifiers, given that it can prove all simpler Σ_1 truths.

We now need to run the same kind of induction again. We first show that $\mathsf{I}\Sigma_1$ can prove $\psi \to \Box_T\psi$ for the simplest Σ_1-wffs ψ. Then we show that $\mathsf{I}\Sigma_1$ can prove $\psi \to \Box_T\psi$ for more complex Σ_1-wffs so long as it can prove all the instances for simpler Σ_1-wffs. This is messy work but not particularly difficult: again, we need to dig inside the box – i.e. inside $\exists \mathsf{v}\mathsf{Prf}_T(\mathsf{v}, \mathsf{x})$ – and rely on the fact that Prf_T recapitulates the definition of the p.r. relation Prf_T. Again we'll not get bogged down by following through all the details here.[2] ⊠

Assuming that our argument-strategy really can be made good, that completes the proof of Theorem 26.1, and hence establishes Theorem 26.2.

26.4 Useful corollaries

That's the main business of this chapter done: but we'll add another three sections (which can be skimmed or skipped). First, in this section, we'll briskly mention some useful corollaries of our arguments so far. In the next section, we'll say something about the Second Theorem for systems of arithmetic that are even weaker than $\mathsf{I}\Sigma_1$. Finally, in the last section, we'll explain a slightly different way of proving the Second Theorem.

The fact that $\mathsf{I}\Sigma_1$ suffices to establish all three derivability conditions for any T extending Q means that we can improve some earlier results in easy ways. Let's highlight just three cases for future reference.

To keep things clear, we'll make subscripts explicit. Then Theorem 25.3 tells us that, if T is nice and satisfies the derivability conditions, $T \vdash \neg\Box_T\varphi \to \mathsf{Con}_T$ (where φ is any sentence). Second, recall Theorem 25.1, the Formalized First Theorem: $T \vdash \mathsf{Con}_T \to \neg\Box\mathsf{G}_T$. Third, Theorem 25.4 tells that that $T \vdash \mathsf{Con}_T \leftrightarrow \mathsf{G}_T$. We can now sharpen these results:

[2] Those whose mathematical masochism wasn't sated by following up the last footnote can get a fuller story in Boolos (1993, pp. 44–49). Needless to say, Boolos's notation is different again. Alternatively, see Rautenberg (2006, Sec. 7.1).

Theorem 26.5 *If T extends* Q*, then (i)* $\mathsf{I}\Sigma_1 \vdash \neg\Box_T\varphi \to \mathsf{Con}_T$ *for any sentence* φ*, (ii)* $\mathsf{I}\Sigma_1 \vdash \mathsf{Con}_T \to \neg\Box\mathsf{G}_T$*, (iii)* $\mathsf{I}\Sigma_1 \vdash \mathsf{Con}_T \leftrightarrow \mathsf{G}_T$*.*

Proof For the first part, simply re-run our argument for $T \vdash \neg\Box_T\varphi \to \mathsf{Con}_T$, replacing '$T \vdash$' throughout by '$\mathsf{I}\Sigma_1 \vdash$'.

Next, reviewing steps (1) to (10) of the proof of Theorem 25.1, we see that we can again replace each '$T \vdash$' by '$\mathsf{I}\Sigma_1 \vdash$' to get $\mathsf{I}\Sigma_1 \vdash \mathsf{Con}_T \to \neg\Box\mathsf{G}_T$.

For the third part, reviewing the proof leading up to Theorem 20.3, we can see that $\mathsf{I}\Sigma_1 \vdash \mathsf{G}_T \leftrightarrow \neg\Box_T\mathsf{G}_T$ (for $\mathsf{I}\Sigma_1$ can capture the *diag* function). Putting that result together with (ii) gives us one direction of the biconditional (iii): $\mathsf{I}\Sigma_1 \vdash \mathsf{Con}_T \to \mathsf{G}_T$.

As a special case of (i), we also have $\mathsf{I}\Sigma_1 \vdash \neg\Box\mathsf{G}_T \to \mathsf{Con}_T$. Combined with $\mathsf{I}\Sigma_1 \vdash \mathsf{G}_T \leftrightarrow \neg\Box\mathsf{G}_T$ again, that gives us the other direction of the biconditional (iii): $\mathsf{I}\Sigma_1 \vdash \mathsf{G}_T \to \mathsf{Con}_T$. ⊠

26.5 The Second Theorem for weaker arithmetics

(a) We noted that Prf_T is a Σ_1 wff. That led us to conjecture that, if T has at least Σ_1-induction, then T will be able to prove the required facts about Prf_T to make the derivability conditions hold.

But in fact, even full Σ_1-induction is more than we need to get the derivability conditions. What is crucial in handling Prf_T is that T can deal with exponentiation which plays such a crucial role in our arithmetization of syntax. So consider the following theory, known as Elementary Arithmetic, which knows more about exponentiation than does Q but is less powerful that $\mathsf{I}\Sigma_1$:

> EA is Q plus Δ_0-induction plus the axiom that exponentiation is always defined – i.e. the axiom $\forall x\forall y\exists!z\,\mathsf{Exp}(x, y, z)$ where Exp captures the exponentiation function.[3]

If a theory T contains at least EA, then in fact that's enough for the derivability conditions to hold: which gives us one way of sharpening Theorem 26.2.

This isn't just a technical curiosity. As its standard label 'elementary arithmetic' suggests, EA is a very interesting theory in its own right.[4] However, we haven't space to pursue the matter here.

(b) And what about theories even weaker than EA? What about $\mathsf{I}\Delta_0$ – which we defined in Section 10.4, fn. 3. – which is what you get by adding just induction for Δ_0 wffs to Q. Or what about our old friend Q itself?

[3] The new final axiom here implies that it is legitimate to add to EA a function symbol for exponentiation (see Section 12.2, (b)). And that point indicates another way of presenting EA. Simply take Q^+ – the theory mentioned in Section 13.4, fn. 5, which adds the exponential function with its obvious formal recursion equations to Q – and then also add the universal closures of all instances of the Induction Schema for wffs, including those with the new function symbol, which have no more than bounded quantifiers.

[4] For an intriguing discussion of the significance of EA, see Avigad (2003).

Given the weakness of these theories, it will be no surprise at all to learn that the Second Theorem applies here too: i.e. $\mathsf{I}\Delta_0 \nvdash \mathsf{Con}_{\mathsf{I}\Delta_0}$, $\mathsf{Q} \nvdash \mathsf{Con}_{\mathsf{Q}}$. But the proofs for these cases do have to go quite differently, because $\mathsf{I}\Delta_0$, and so Q, isn't strong enough for e.g. the third derivability condition to hold.[5]

26.6 Jeroslow's Lemma and the Second Theorem

(a) Finally in this chapter, we'll introduce a different version of our generalized Second Theorem which is due to R. G. Jeroslow (1973), in part because it is intriguing, in part because it is mentioned from time to time in the secondary literature. Let's start by proving what can be called Jeroslow's Diagonalization Lemma.

Here's the classic version of the Diagonalization Lemma that underlies our previous incompleteness proofs:

> **Theorem 20.4** *If T is a nice theory and $\varphi(\mathsf{x})$ is* any *wff of its language with one free variable, then there is a sentence γ such that $T \vdash \gamma \leftrightarrow \varphi(\ulcorner\gamma\urcorner)$.*

And now here's the variant Lemma due to Jeroslow:

> **Theorem 26.6** *If T is a theory which has a function symbol to capture each p.r. function, and $\varphi(\mathsf{x})$ is* any *wff of its language with one free variable, then there is a term τ such that $T \vdash \tau = \ulcorner\varphi(\tau)\urcorner$.*

Note that the assumption in this theorem is in fact easily satisfied. If a theory contains Q and a smidgin of induction, it can fully capture each p.r. function as a function (it will be strongly p.r. adequate): there will then be a definitional extension of the theory which contains a function symbol for each p.r. function.

Proof Given an open wff $\varphi(\mathsf{x})$, we can define a corresponding diagonalizing function d which takes the g.n. of a function symbol f, and returns the g.n. of the wff $\varphi(\mathsf{f}(\ulcorner\mathsf{f}\urcorner))$ (for arguments which aren't codes for monadic function symbols, the value of d can default to zero). This function is evidently p.r. because its value can be computed without unbounded loopings.

Since d is p.r., it is captured in T by some function symbol d. Hence by its definition, $d(\ulcorner\mathsf{d}\urcorner) = \ulcorner\varphi(\mathsf{d}(\ulcorner\mathsf{d}\urcorner))\urcorner$. And therefore, by the definition of capturing, $T \vdash \mathsf{d}(\ulcorner\mathsf{d}\urcorner) = \ulcorner\varphi(\mathsf{d}(\ulcorner\mathsf{d}\urcorner))\urcorner$. Put $\tau = \mathsf{d}(\ulcorner\mathsf{d}\urcorner)$ and Jeroslow's Lemma is proved. ⊠

(b) For a preliminary application of this Lemma, take the instance of our new theorem where $\varphi(\mathsf{x}) =_{\text{def}} \neg\mathsf{Prov}(\mathsf{x})$. So, assuming T satisfies the condition in the Lemma, there's a term τ such that

[5] See Wilkie and Paris (1987) and Bezboruah and Shepherdson (1976) respectively for the proofs of the Second Theorem for $\mathsf{I}\Delta_0$ and Q. See Willard (2001) for an extended exploration of some hobbled and perhaps not-very-natural arithmetics which *can* prove (some of) their own consistency sentences.

1. $T \vdash \tau = \ulcorner\neg\mathsf{Prov}(\tau)\urcorner$

Now define the sentence $\mathsf{G}' =_{\text{def}} \neg\mathsf{Prov}(\tau)$, and let's suppose

2. $T \vdash \mathsf{G}'$

That means that for some number m, $Prf(m, \ulcorner\neg\mathsf{Prov}(\tau)\urcorner)$ is true (just by the definition of the *Prf* relation). Hence, since Prf captures *Prf* in T, we have

3. For some m, $T \vdash \mathsf{Prf}(\mathsf{m}, \ulcorner\neg\mathsf{Prov}(\tau)\urcorner)$.

Whence, using (1), we get

4. For some m, $T \vdash \mathsf{Prf}(\mathsf{m}, \tau)$.

But since G' is equivalent to $\forall\mathsf{v}\neg\mathsf{Prf}(\mathsf{v}, \tau)$, (4) contradicts (2). So, given that T is consistent, (2) must be false: T cannot prove G'. Similarly, we can show that T cannot prove $\neg\mathsf{G}'$, on pain of being ω-inconsistent (exercise: check that claim). In sum, a version of the First Theorem quickly follows again from Jeroslow's version of the Diagonalization Lemma.

(c) That was mildly diverting, but doesn't tell us anything novel. However, we'll now use Jeroslow's Lemma to derive a version of the Second Theorem, and this time there *is* a new twist.

First, recall that '$\dot\neg\mathsf{x}$' captures the function which maps the g.n. of a wff φ to the g.n. of its negation: thus $\dot\neg\ulcorner\varphi\urcorner = \ulcorner\neg\varphi\urcorner$ (see Section 21.3, fn. 4 for more explanation). And, as we noted in Section 24.5, one natural way of expressing a theory's consistency is

$$\mathsf{Con}'_T =_{\text{def}} \neg\exists\mathsf{x}(\mathsf{Prov}_T(\mathsf{x}) \wedge \mathsf{Prov}_T(\dot\neg\mathsf{x}))$$

Let's suppose for reductio that

1. $T \vdash \mathsf{Con}'_T$

And now take the wff $\varphi(\mathsf{x})$ in Jeroslow's Lemma to be $\mathsf{Prov}_T(\dot\neg\mathsf{x})$. So, dropping subscripts again, for some term τ,

2. $T \vdash \tau = \ulcorner\mathsf{Prov}(\dot\neg\tau)\urcorner$

We can then argue as follows, assuming that conditions (C1) and (C3) apply to T and that T has some basic logic.

3.	$T \vdash \mathsf{Prov}(\dot\neg\tau) \rightarrow \mathsf{Prov}(\dot\neg\tau)$	Logic!
4.	$T \vdash \mathsf{Prov}(\dot\neg\tau) \rightarrow \mathsf{Prov}(\dot\neg\ulcorner\mathsf{Prov}(\dot\neg\tau)\urcorner)$	From 2, 3
5.	$T \vdash \mathsf{Prov}(\dot\neg\tau) \rightarrow \mathsf{Prov}(\ulcorner\mathsf{Prov}(\dot\neg\tau)\urcorner)$	Instance of C3
6.	$T \vdash \mathsf{Prov}(\dot\neg\tau) \rightarrow$ $(\mathsf{Prov}(\ulcorner\mathsf{Prov}(\dot\neg\tau)\urcorner) \wedge \mathsf{Prov}(\dot\neg\ulcorner\mathsf{Prov}(\dot\neg\tau)\urcorner))$	From 4 and 5
7.	$T \vdash \mathsf{Prov}(\dot\neg\tau) \rightarrow \neg\mathsf{Con}'$	From 6, defn. Con'
8.	$T \vdash \neg\mathsf{Prov}(\dot\neg\tau)$	From 1, 7

9.	$T \vdash \mathsf{Prov}(\ulcorner \neg \mathsf{Prov}(\dot{\neg}\tau) \urcorner)$	From 8, by C1
10.	$T \vdash \mathsf{Prov}(\dot{\neg} \ulcorner \mathsf{Prov}(\dot{\neg}\tau) \urcorner)$	Defn. of $\dot{\neg}$ fn.
11.	$T \vdash \mathsf{Prov}(\dot{\neg}\tau)$	From 2, 10
12.	Contradiction!	From 8, 11.

So we can conclude that (1) can't be true: $T \nvdash \mathsf{Con}'_T$.[6]

(d) Which all goes to establish the following theorem (assuming T has the modicum of logic needed in the argument above):

> **Theorem 26.7** *If T proves the Jeroslow Diagonalization Lemma, and the derivability conditions (C1) and (C3) hold, then $T \nvdash \mathsf{Con}'_T$.*

Hence (C2) *doesn't* always have to hold for a provability predicate Prov_T for the corresponding sentence Con'_T to be unprovable in T: it isn't an *essential* derivability condition for a version of the Second Theorem to apply.

(e) But is this tweaked result of much interest? Usually not! For we've already seen that in any normal theory with a standard first-order logic and which includes the weak theory EA, all three derivability conditions *will* hold for the canonical proof predicate. So Jeroslow's result only really kicks in when we are either dealing with extraordinary theories (e.g. with a cut-down logic), or with extraordinary provability predicates (about which a little more in the next chapter).

However, it has been quite fun to see Jeroslow's neat trick for finding a new version of the Diagonalization Lemma which uses identity rather than the biconditional. And it justifies the following decidedly neat summary thought:

> What it fundamentally takes to prove the *First* Incompleteness Theorem is that we are dealing with a theory which is Σ_1-complete; if the theory *knows* that it is Σ_1-complete then the *Second* Theorem applies.

Why? Well, Σ_1-completeness is the basic ingredient in our proof that Q and richer nice theories T capture all the p.r. functions and relations. And that fact in turn underpins the argument for the First Incompleteness Theorem. Now, further, if T is Σ_1-complete, then (C1) holds by the argument in Section 20.1. And for T to 'know' it is Σ_1-complete is for T to be able to prove $\psi \to \Box\psi$ for every Σ_1 sentence ψ, and that as we noted in Section 26.3 is enough for (C3) to hold too. But Jeroslow has now shown us that – so long as we are dealing with a theory with function symbols for every p.r. function – that's enough for a version of the Second Theorem to hold.

[6] An exercise to check your understanding: why haven't we made our argument look prettier by using our abbreviatory $\Box$ symbol?

27 Reflections

After the glory days of the 1930s, Gödel's comments on the details of his incompleteness theorems were few and far between. However, he did add a brief footnote to the 1967 translation of a much earlier piece on 'Completeness and consistency'. And Gödel thought that his brisk remarks in that footnote were sufficiently important to repeat them in a short paper in 1972, in a section entitled 'The best and most general version of the unprovability of consistency in the same system'.[1]

Gödel makes two main points. We explain the first of them in Section 27.2. We then go on to prove some results about reflection principles which hopefully throw light on his second point. And we'll return to develop that second point further in the next chapter, where we touch on Hilbert's Programme.

It will do no harm at all, however, to begin with a summary of ...

27.1 The Second Theorem: the story so far

(a) Start with the p.r. relation $Prf_T(m, n)$, which obtains when m is the super g.n. of a T-proof of the wff with g.n. n. Assuming T is p.r. adequate, this relation can be *canonically captured* in T by an open wff $\mathsf{Prf}_T(\mathsf{x}, \mathsf{y})$ whose components recapitulate step by step the natural p.r. definition of Prf_T (along the lines we gave for the case of PA back in Section 15.9). Of course, Prf_T won't be unique (see Section 13.8): there can be minor variations among natural p.r. definitions of Prf_T, and then we can always re-order conjuncts and change variables when we come to form Prf_T. We needn't fuss about such differences.

Given Prf_T, we then define a corresponding proof predicate $\mathsf{Prov}_T(\mathsf{x}) =_{\text{def}} \exists \mathsf{v}\mathsf{Prf}_T(\mathsf{v}, \mathsf{x})$. And we can use this proof predicate to define a number of natural consistency sentences for T, Con_T, Con'_T, Con''_T, which 'indirectly say' respectively that T can't prove absurdity, that T can't prove a contradictory pair of wffs, and that T can't prove every sentence. In normally set-up theories with a classical logic, these consistency sentences will be provably equivalent so we again needn't fuss about the differences.

A nice theory, recall, is one that is consistent, p.r. axiomatized, and contains Q. Now define a nice* theory to be one which is nice and which also includes induction for (at least) Σ_1 wffs. Nice* theories are strong enough for the so-called derivability conditions to apply to the canonical provability predicate Prov_T. In fact, being nice* is rather more than enough, as we noted in Section 26.5: but

[1] The original note 'Completeness and consistency' is Gödel (1932). The added material first appears in van Heijenoort (1967, pp. 616–617) and is then reused in Gödel (1972).

that's one more thing we aren't going to fuss about. And when the derivability conditions apply, T can't prove Con_T. Which gives us the generalized Second Theorem in the form of Theorem 26.2: if T is nice*, then $T \nvdash \mathsf{Con}_T$.

(b) But now let's emphasize again that, in showing that the derivability conditions hold for Prov_T, we have to rely on the fact that Prov_T is defined in terms of a predicate expression Prf_T which *canonically* captures Prf_T. Putting it roughly, we need $\mathsf{Prf}_T(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ to reflect the details of what it takes for m to code a proof of the wff with g.n. n. Putting it even more roughly, we need Prf_T to have the right intended *meaning*.

Recall our discussion in Section 4.6 (and also cf. Section 16.8). If $\mathsf{Prf}_T(\mathsf{x}, \mathsf{y})$ captures Prf_T, then so does $\mathsf{Prf}_T(\mathsf{x}, \mathsf{y}) \wedge \theta$, for any and every T-theorem θ; but in most cases, such a wff won't canonically capture Prf_T. Likewise for most wffs $\mathsf{Prf}^\dagger_T$ which capture the desired relation; these too won't canonically capture it. And while the derivability conditions hold for $\mathsf{Prov}_T =_{\text{def}} \exists \mathsf{v} \mathsf{Prf}_T(\mathsf{v}, \mathsf{x})$ (at least if T has a smidgin of induction), they in general won't hold for the corresponding predicates $\mathsf{Prov}^\dagger_T(\mathsf{x}) =_{\text{def}} \exists \mathsf{v} \mathsf{Prf}^\dagger_T(\mathsf{v}, \mathsf{x})$. Hence, we won't necessarily be able to derive the corresponding results that, if T is nice*, then $T \nvdash \mathsf{Con}^\dagger_T$ (where $\mathsf{Con}^\dagger_T$ is defined in the obvious way from $\mathsf{Prov}^\dagger_T$).

(c) So, in summary, there's an important contrast between what it takes to prove the First Theorem and the Second Theorem.

Take *any* old wff $\mathsf{Prf}^\dagger_T$ which captures Prf_T; form the corresponding proof predicate $\mathsf{Prov}^\dagger_T$; take a fixed point γ for $\neg \mathsf{Prov}^\dagger{}_T(\mathsf{x})$; and we can show $T \nvdash \gamma$ (assuming T is nice).

By contrast, if we want to prove $T \nvdash \mathsf{Con}_T$ (assuming T is nice*), we have to be more picky. We need to start with a wff Prf_T which 'has the right meaning', which *canonically* captures Prf_T, and then form the canonical consistency sentence Con_T from *that*.[2]

27.2 There are provable consistency sentences

(a) Those last remarks raise again a question which we left hanging at the end of Section 24.5, which is also touched on by Gödel in his 1967/1972 note. So let's now pause over it. Can there be a *non-canonical* sentence $\mathsf{Con}^\dagger_T$ – a sentence built from a non-canonical proof predicate – which in some fairly natural way still 'says' that T is consistent but which *is* provable in T?

The answer is 'yes'. As Gödel puts it in his note:

> ... the consistency (in the sense of non-demonstrability of both a proposition and its negation), even of very strong systems S, *may* be provable in S. (Gödel, 1972, p. 305)

[2] Feferman (1960) – perhaps a bit dangerously – describes results like the Second Theorem which depend on Prf_T 'more fully express[ing] the notion involved' as 'intensional'.

There are a number of possibilities, using various non-canonical consistency sentences. We'll concentrate here on the construction that – at least at the outset – looks least like a cheap trick.

(b) Start with the following thought. When trying to establish an as-yet-unproved conjecture, mathematicians will use any tools to hand, bringing to bear any background assumptions that they are prepared to accept in the context.[3] The more improvisatory the approach, the less well-attested the assumptions, then the greater the risk of lurking inconsistencies emerging, requiring our working assumptions to be revised. A check needs to be kept that apparent new results cohere with secure background knowledge. Only a derivation which passes the coherence test has a chance of being accepted as a *proof*.

Here, then, is a possible way of capturing just something of the idea that a genuine proof should involve checking consistency with what's gone before. Say that ψ simply contradicts φ if one is the negation of the other. Then there is a *consistency-minded proof* of φ in the formal system T iff (i) there is an ordinary T-derivation of φ with super g.n. m, while (ii) there isn't already a T-derivation with a code number less than m of a sentence which simply contradicts φ.

So, take the numerical relation *Contr* defined as follows:

i. $Contr(m, n) =_{\text{def}} (m = \ulcorner\neg\urcorner \star n \ \vee \ n \ = \ulcorner\neg\urcorner \star m)$.

This is the p.r. relation which holds between two numbers when one codes for the negation of the wff coded for by the other. And we can use this to define the relation

ii. $CPrf_T(\mathsf{x}, \mathsf{y}) =_{\text{def}} Prf(x, y)_T \wedge \neg(\exists u \leq x)(\exists w \leq u)(Prf_T(u, w) \wedge Contr(y, w))$

where, to give a nice bound to the second quantifier, we rely on the fact that on our coding scheme, if u codes for a proof of the wff with g.n. w, then $w < u$. Evidently, $CPrf_T(m, n)$ holds when m numbers an ordinary T-proof of φ and there is no earlier proof of a wff which simply contradicts φ. In other words, m codes for a consistency-minded proof of the sentence numbered n.

The p.r. relation *Contr* can, of course, be captured in a nice theory by a corresponding wff $\mathsf{Contr}(\mathsf{x}, \mathsf{y})$. So if we define

ii. $\mathsf{CPrf}_T(\mathsf{x}, \mathsf{y}) =_{\text{def}} \mathsf{Prf}(\mathsf{x}, \mathsf{y})_T \wedge \neg(\exists \mathsf{u} \leq \mathsf{x})(\exists \mathsf{w} \leq \mathsf{u})(\mathsf{Prf}_T(\mathsf{u}, \mathsf{w}) \wedge \mathsf{Contr}(\mathsf{y}, \mathsf{w}))$.

this wff will in fact capture the p.r. relation $CPrf_T$. And now, just as we defined the provability predicate Prov_T in terms of Prf_T, we can define the consistency-minded provability predicate

iii. $\mathsf{CProv}_T(\mathsf{x}) =_{\text{def}} \exists \mathsf{v}\, \mathsf{CPrf}_T(\mathsf{v}, \mathsf{x})$.

[3] See the classic Lakatos (1976) for a wonderfully lively exploration of the process of the growth of mathematical knowledge.

$\mathsf{CProv}_T(\ulcorner\varphi\urcorner)$ will therefore be true just so long as there is a consistency-minded proof of φ.

(c) Two remarks about our definitions. First, CProv_T is, of course, a *very* close cousin of RProv_T, the Rosser provability predicate which we met in Section 21.3. But Rosser provability is, so to speak, lopsidedly consistency-minded: φ is only Rosser-provable so long as there is no 'smaller' proof of $\neg\varphi$ (but $\neg\varphi$ can be Rosser-provable even if there is a 'smaller' proof of plain φ). By contrast, the property expressed by CProv_T is *symmetrically* consistency-minded.

Second, suppose we are dealing with a theory T which actually *is* consistent: then every T-proof is de facto consistency-minded. In this case, the relation $CPrf_T$ is extensionally the same relation as Prf_T, so CPrf_T captures Prf_T too. Moreover, $\mathsf{CProv}_T(\overline{\mathsf{n}})$ will be true just when $\mathsf{Prov}_T(\overline{\mathsf{n}})$ is true. So, for a consistent theory, CProv_T *in fact* expresses the same familiar provability property as Prov_T – or at any rate, it expresses the same property by our official extensional standards. Let's say, though, that CProv_T expresses this *obliquely*, because the predicate doesn't, as it were, reveal on its face that it *is* expressing just that property.

(d) So, at least at first sight, there seems to be some interest in the consistency-minded proof-predicate CProv_T. Let's now put

iv. $\mathsf{CCon}_T =_{\text{def}} \neg\exists\mathsf{x}\exists\mathsf{y}(\mathsf{CProv}_T(\mathsf{x}) \wedge \mathsf{CProv}_T(\mathsf{y}) \wedge \mathsf{Contr}(\mathsf{x},\mathsf{y}))$.

This might seem to be a natural enough way of defining a consistency sentence in terms of consistency-minded provability. Note again that, if the theory we are dealing with *is* consistent, then since CProv_T in such a case obliquely expresses the property of plain provability, then we might say that CCon_T obliquely expresses the claim that the theory is consistent.

(e) And now for the result we've been working up to. It is quite straightforward to show that

> **Theorem 27.1** *For any T, $\mathsf{I}\Sigma_1 \vdash \mathsf{CCon}_T$; hence for any nice* T, $T \vdash \mathsf{CCon}_T$.*

Proof sketch Argue inside $\mathsf{I}\Sigma_1$ and let's drop subscript T's for readability. We'll suppose $\neg\mathsf{CCon}$, that is to say $\exists\mathsf{x}\exists\mathsf{y}(\mathsf{CProv}(\mathsf{x}) \wedge \mathsf{CProv}(\mathsf{y}) \wedge \mathsf{Contr}(\mathsf{x},\mathsf{y}))$, and derive a contradiction.

For some c and d, therefore, we have $\exists\mathsf{v}\,\mathsf{CPrf}(\mathsf{v},\mathsf{c}) \wedge \exists\mathsf{v}\,\mathsf{CPrf}(\mathsf{v},\mathsf{d}) \wedge \mathsf{Contr}(\mathsf{c},\mathsf{d})$. Hence, for some a, b, $\mathsf{CPrf}(\mathsf{a},\mathsf{c}) \wedge \mathsf{CPrf}(\mathsf{b},\mathsf{d}) \wedge \mathsf{Contr}(\mathsf{c},\mathsf{d})$.

However, Σ_1-induction is (more than) enough to prove $\forall\mathsf{x}(\mathsf{x} \leq \mathsf{y} \vee \mathsf{y} \leq \mathsf{x})$ – see Section 10.3. Hence we can prove that $(\mathsf{a} \leq \mathsf{b} \vee \mathsf{b} \leq \mathsf{a})$.

Now argue by cases. Let $\mathsf{a} \leq \mathsf{b}$. By hypothesis, we have $\mathsf{CPrf}(\mathsf{a},\mathsf{c})$; which by definition implies $\mathsf{Prf}(\mathsf{a},\mathsf{c})$. But by construction of Prf, that in turn implies $\mathsf{c} \leq \mathsf{a}$. Put together our two ordering claims with $\mathsf{Prf}(\mathsf{a},\mathsf{c})$ and $\mathsf{Contr}(\mathsf{c},\mathsf{d})$ and we can derive $(\exists\mathsf{u} \leq \mathsf{b})(\exists\mathsf{w} \leq \mathsf{u})(\mathsf{Prf}(\mathsf{u},\mathsf{w}) \wedge \mathsf{Contr}(\mathsf{w},\mathsf{d}))$. But by definition $\mathsf{CPrf}(\mathsf{b},\mathsf{d})$ implies the negation of that. Contradiction.

In the same way, $\mathsf{b} \leq \mathsf{a}$ also leads to contradiction. So we are done: our initial supposition leads through to contradictions so the first part of the theorem is proved. The second part of the theorem then follows trivially. ⊠

But $\mathsf{I}\Sigma_1$ is of course a sound theory; so CCon_T is always true, for *any* T. Which, as we'll see in a moment, immediately deprives its provability of any interest!

27.3 What does that show?

Gödel is right. Suppose T is nice*. Then, being consistent, all T's proofs are consistency-minded. And it *can*, in particular, prove a sentence which – in a slightly oblique way – claims the 'non-demonstrability of both a proposition and its negation' in that system itself. But now note the following two points:

(a) Whether φ has a consistency-minded proof in some given theory T doesn't depend just on the axioms and basic rules of inference of T, but also on the quite extraneous question of how we order derivations in T by their Gödel numbers. Take an ordinarily inconsistent theory T; then on one numbering of proofs, there could be a consistency-minded T-proof of φ but not of $\neg\varphi$ (the derivation of φ has the lower g.n. on the chosen numbering scheme), while on another equally legitimate numbering system, there could be a consistency-minded T-proof of $\neg\varphi$ but not of φ. But a notion of 'proof' where what can be demonstrated by that kind of proof depends on a quite arbitrary choice, entirely extraneous to the axioms and basic rules of inference (and not settled by any considerations of truth and truth-transmission), is surely not a notion of proof worth having.

(b) More fundamentally, the truth of CCon_T tells us nothing about T (for as we said, that sort of wff is true indiscriminately, for any T). For example, it is perfectly possible that there are consistency-minded proofs in T of φ, $\varphi \to \psi$, and $\neg\psi$. That's because the smallest-numbered putative T-derivation of ψ – e.g. the result of chaining a proof of φ after a proof of $\varphi \to \psi$ and then using modus ponens – could have a g.n. greater than a proof of $\neg\psi$.[4]

Which goes to show that the truth of CCon_T leaves wide open the possibility that T's set of consistency-minded theorems is *semantically* inconsistent (i.e. there is no interpretation respecting the meaning of the logical apparatus which makes those consistency-minded theorems all true).

Hence, to take an extreme example, showing that $\mathsf{I}\Sigma_1 \vdash \mathsf{CCon}_{\mathsf{ZFC}}$ and – therefore that $\mathsf{CCon}_{\mathsf{ZFC}}$ is true – goes no way at all towards settling whether the standard set theory ZFC is semantically consistent. Yet semantic consistency is

[4]Since $T \vdash \mathsf{CCon}_T$, we know that the derivability conditions can't all hold for the consistency-minded provability predicate CProv_T (else we'd be able to show $T \nvdash \mathsf{CCon}_T$). We now see that, for a start, the condition (C2) can't hold.

Let's be clear about this. CPrf_T captures the relation $CPrf_T$; and if T is consistent, that means it captures Prf_T. But it doesn't capture Prf_T *canonically*. Because of that, the predicate CProv_T defined in terms of CPrf_T can fail to satisfy the derivability conditions.

surely what we really most care about in seeking proofs of consistency in the first place.

So: even though the initial motivating idea was not implausible, it has turned out that proving 'symmetrically Rosserized' consistency sentences doesn't settle anything. So we won't discuss this or related bits of trickery any further.[5]

27.4 The reflection schema: some definitions

Our next main topic concerns what happens if you add instances of a reflection schema to a theory.

We need to introduce some jargon for use over the coming few sections; so let's gather the various definitions we need into one place. We start with four generic definitions.

i. As we said in Section 25.7, the *reflection schema* for a theory T is

$$\Box_T\varphi \to \varphi$$

where φ holds the place for any sentence of T's language. For example, if we put '$1 + 1 = 2$' for φ, then the corresponding instance of the schema says that if T proves '$1 + 1 = 2$', then $1 + 1 = 2$.

ii. We'll also say that the Π_1 *reflection schema* for T is the reflection schema restricted to cases where φ is a Π_1 sentence of T's language.

iii. A theory S *proves* Π_1 *reflection for* T iff, for every Π_1 sentence φ belonging to T's language, $S \vdash \Box_T\varphi \to \varphi$.

iv. Following our definition at the beginning of Section 9.7, we'll say that a theory T is Π_1*-sound* when, for every Π_1 sentence φ, if $T \vdash \varphi$ then φ is true. It is immediate that a theory T is Π_1-sound when every instance of its Π_1 reflection schema is true.

Now we add two more specific definitions, concerning ways to extend the theory PA:

v. The theory PAC is the theory you get by adding PA's canonical consistency sentence Con as a new axiom to PA.

vi. The theory PAΠ is the theory you get by adding to PA all instances of its Π_1 reflection schema as additional axioms.

In the next section we'll work up to a proof that PAC and PAΠ are equivalent theories (i.e. every theorem of one is a theorem of the other).

[5]The idea of using a Rosserized predicate like CProv_T to form a provable consistency statement seems to be due to Kreisel. He mentions it in passing in Kreisel (1965, p. 154), and there are some more comments in his quirky but illuminating memoir of Gödel (Kreisel, 1980, p. 173). Feferman's introduction in Gödel (1990, pp. 282–287) also notes some more ways in which theories can in one sense or another prove their own consistency.

For a more advanced discussion, see e.g. Visser (1989).

27.5 Reflection and PA

(a) We can slightly sharpen Theorem 25.7:

> **Theorem 27.2** *If T is nice, then T cannot prove all instances of the Π_1 reflection schema for T.*

Proof G_T is Π_1 and so if T proved all instances of its Π_1 reflection schema, then $T \vdash \Box\mathsf{G} \to \mathsf{G}$. But, being nice, $T \vdash \mathsf{G} \leftrightarrow \neg\Box\mathsf{G}$. From which it would follow that $T \vdash \mathsf{G}$ contrary to the first incompleteness theorem. ⊠

This result applies to PA in particular. However, if we believe that PA is sound – as we surely do! – then we'll believe that whatever it proves is true. So we'll think that every instance of its reflection schema is in fact *true*. In particular the instances of PA's Π_1 reflection schema are all true. So let's consider the extended theory PAΠ which we get by adopting all those instances as further axioms.

That's still a p.r. axiomatized theory, since we can test whether a wff is one of the new axioms without an unbounded search. Given that PA is sound, PAΠ is also sound, so must be consistent.[6] So PAΠ is another nice theory, assuming PA is.

(b) We'll now show that, still assuming PA is nice, PAΠ proves Con, the consistency sentence for PA, and hence proves PA's canonical Gödel sentence G. (For the rest of this section, the unadorned box indicates provability in PA.)

Proof $\bot$ is Π_1 (why?); so since PAΠ proves all instances of PA's Π_1 reflection schema, PA$\Pi \vdash \Box\bot \to \bot$, hence PA$\Pi \vdash \neg\Box\bot$. In other symbols, PA$\Pi \vdash$ Con.

But we've shown that, being nice, PA $\vdash$ Con $\to$ G (see the proof for Theorem 25.2). Hence PA$\Pi \vdash$ Con $\to$ G. Hence PA$\Pi \vdash$ G. ⊠

(c) Conversely: the theory PAC which has Con as an additional axiom proves all instances of Π_1 reflection for PA:

Proof PAC extends $\mathsf{I}\Sigma_1$ and the derivability conditions apply. Suppose φ is some Π_1 sentence. Then $\neg\varphi$ is Σ_1, so we can apply Theorem 26.4 to this wff, to get

1.	PAC $\vdash \neg\varphi \to \Box\neg\varphi$	
2.	PAC $\vdash \neg\varphi \to (\varphi \to \bot)$	Logic
3.	PAC $\vdash \Box(\neg\varphi \to (\varphi \to \bot))$	From 2, by C1
4.	PAC $\vdash \Box\neg\varphi \to \Box(\varphi \to \bot)$	From 3, using C2
5.	PAC $\vdash \Box(\varphi \to \bot) \to (\Box\varphi \to \Box\bot)$	By C2
6.	PAC $\vdash \neg\varphi \to (\Box\varphi \to \Box\bot)$	From 1, 4, 5
7.	PAC $\vdash \neg\Box\bot$	Con is an axiom!
8.	PAC $\vdash \Box\varphi \to \varphi$	From 6, 7

[6] NB: Löb's Theorem shows that a nice theory T can't prove all the instances of T's *own* reflection schema: there's nothing to stop a richer theory T' proving all the instances of T's reflection schema.

Since φ was an arbitrary Π_1 sentence, that shows that PAC proves any instance of Π_1 reflection for PA. So we are done. ⊠

(d) In sum, we've just shown that the result of adding to PA all the instances of Π_1 reflection entails Con; and the result of adding Con to PA entails each instance of Π_1 reflection for PA. Which evidently is enough to establish the promised result:

Theorem 27.3 *The theories* PAΠ *and* PAC *are equivalent.*

27.6 Reflection, more generally

It will be no surprise at all to learn that we can now generalize our results linking the provability of Con and the provability of Π_1 reflection.

Recall our neat little theorem:

Theorem 9.4 *If T extends* Q*, T is consistent iff it is Π_1-sound.*

So this tells us, equivalently, that for theories which extend Q, Con_T is true iff all the instances of its Π_1 reflection schema are true. And now we can formally prove that fact inside suitable theories:

Theorem 27.4 *Suppose T extends* Q*, and suppose that the theory S extends $\mathsf{I}\Sigma_1$: then $S \vdash \mathsf{Con}_T$ iff S proves Π_1 reflection for T.*

Proof Just generalize the arguments in the last section. For the 'if' direction, suppose S proves Π_1 reflection for T, i.e. for every Π_1 sentence φ, $S \vdash \Box_T\varphi \to \varphi$. Then in particular, $S \vdash \Box_T\bot \to \bot$. Whence $S \vdash \neg\Box_T\bot$, i.e. $S \vdash \mathsf{Con}_T$.

For the 'only if' direction, suppose φ is a Π_1 sentence. We note that we can argue as in (d) in the last section to get as far as

6. $S \vdash \neg\varphi \to (\Box_T\varphi \to \Box_T\bot)$

since S contains $\mathsf{I}\Sigma_1$ (and that's enough to work with T's derivability conditions). Then we can continue

7.	$S \vdash \neg\Box_T\bot$	By hypothesis
8.	$S \vdash \Box_T\varphi \to \varphi$	From 6, 7

So we are done. ⊠

Now, by Theorem 26.5, $\mathsf{I}\Sigma_1 \vdash \mathsf{Con}_T \leftrightarrow \mathsf{G}_T$. So our new theorem again links the provability of Π_1 reflection for T to the provability of G_T. But in fact we can easily make this connection more directly:

Theorem 27.5 *If S extends* Q *and proves Π_1 reflection for T, then $S \vdash \mathsf{G}_T$.*

Proof $\mathsf{Q} \vdash \mathsf{G}_T \leftrightarrow \neg\Box\mathsf{G}_T$, since the proof of Theorem 20.3 calls on no more than the resources of Q (why?). Hence $S \vdash \mathsf{G}_T \leftrightarrow \neg\Box\mathsf{G}_T$ too. By hypothesis, S proves Π_1 reflection for T; but G_T is Π_1; so $S \vdash \Box\mathsf{G}_T \to \mathsf{G}_T$. But then $S \vdash \Box\mathsf{G}_T \to \neg\Box\mathsf{G}_T$, so $S \vdash \neg\Box\mathsf{G}_T$, and therefore $S \vdash \mathsf{G}_T$. ⊠

27.7 'The best and most general version'

Suppose we are interested in Hilbert's project of trying to use uncontentious reasoning to show that the wilder reaches of mathematics are still 'safe' (see Section 1.6). What does 'safety' consist in? Well, you might very reasonably suppose that one condition any acceptable theory T should satisfy is this: it shouldn't actually get things wrong about the Π_1 truths of arithmetic. After all, those are essentially just the true equations involving particular numbers, and universally quantified versions of such equations. So we'll certainly want it to be the case that if $T \vdash \varphi$ and φ is Π_1, then φ is true.

Thus, being Π_1-sound looks to be a natural minimal condition for being a 'safe' theory. Gödel calls in effect this same condition 'outer consistency' (though the label hasn't caught on). And he remarks that 'for the usual systems [outer consistency] is trivially equivalent with consistency', which is just Theorem 9.4 again (Gödel, 1972, p. 305).

To establish formally that a theory T is Π_1-sound and hence to that extent 'safe' is a matter of proving Π_1 reflection for T. But Theorem 27.2 tells us that a nice T can't prove Π_1 reflection for itself, and it follows that an arithmetic like PA certainly can't prove Π_1 reflection for a stronger, more infinitary, theory T either, unless T is inconsistent and proves everything. *In sum, we can't use relatively modest arithmetical reasoning to prove even the* Π_1*-soundness of a (consistent) infinitary theory* T*.*

As we'll see in the following Interlude, if we are interested in the Hilbertian project of certifying the safety of some infinitary theory T, then this result that we can't establish T's Π_1-soundness using non-infinitary reasoning is arguably the crucial one. It is already enough to undermine the project. Yet note that the unprovability of 'outer consistency', as Gödel calls it, is an easier result than the unprovability of 'inner consistency', i.e. the unprovability of Con_T. For the former result holds given that the Diagonalization Lemma and hence the First Theorem hold for T. So, to show the unprovability of *outer* consistency/Π_1-soundness for T, we don't have to do the hard work of showing that the derivability conditions hold. By contrast, showing the unprovability of *inner* consistency is tougher.

Which is why the result that a theory T can't prove Π_1 reflection for T (if the conditions for the First Theorem apply to T) might be said to be 'the best and most general version of the unprovability of consistency in the same system'.

And that is almost, but not quite, what Gödel says in his 1967/1972 note. What he actually asserts to be the best result is this (with a trivial change):

> [W]hat can be shown to be unprovable in T is the fact that the rules of the equational calculus applied to equations demonstrable in T between primitive recursive terms yield only correct numerical equations (provided only that T possesses the property which is asserted to be unprovable).

Here Gödel's T is a properly axiomatized theory which includes enough arithmetic

to be p.r. adequate. But 'equations between primitive recursive terms' are expressions of the form $\forall x(\mathsf{f}x = \mathsf{g}x)$, where $\mathsf{f}x$ and $\mathsf{g}x$ express p.r. functions, so $\mathsf{f}x = \mathsf{g}x$ expresses a p.r. property. And, assuming T contains enough arithmetic to know about p.r. functions, such expressions will be provably equivalent to Π_1 sentences (cf. Section 16.4).

So what Gödel is saying is in effect that T can't prove Π_1 reflection for T. And he claims that the key condition under which this holds is that Π_1 reflection is actually true for T, i.e. T is Π_1-sound, which – given T includes Q – is equivalent to T's being consistent. But T's being consistent is just what is essential to the First Theorem holding for T (given that T is properly axiomatized and contains enough arithmetic). Hence Gödel's remark seems to be making a version of our point above: so long as the conditions for the First Theorem applying to T hold, T can't prove Π_1 reflection for T.[7]

27.8 Another route to accepting a Gödel sentence?

We'll return to say more about Hilbert's Programme in the next chapter. But before moving on, let's pause to consider whether our recent discussions throw any further light on the question we raised in Section 18.3: what routes are there that can lead us to accept a canonical Gödel sentence G_T as true?

We know that *one* kind of route goes via an explicit judgement that T is consistent: and we stressed in our discussion before that there is quite a variety in the reasons we might have for forming that judgement. But now let's ask: do we *have* to go via explicit reflections about consistency? Can we perhaps go instead via thoughts about Π_1 reflection and rely on results such as that $\mathsf{PA}\Pi \vdash \mathsf{G}$?

Let me spin a just-so story. Imagine someone (we'll call him Kurt) who is a devotee of mathematical rigour and who delights in regimenting his knowledge into systems of Bauhaus austerity. Kurt's explorations, let's suppose, lead him to work within PA as an elegantly neat framework in which he can deduce all the familiar facts about the basic arithmetic of addition and multiplication, and lots of the less familiar facts too. Moreover, Kurt discovers the β-function trick which allows him to introduce definitions for all the *other* p.r. functions he knows about; so he can deduce what he knows about those p.r. functions too. Thoroughly immersed in the theory, Kurt enthusiastically follows deductions in PA wherever they take him: whenever he can derive φ in PA, then he adds φ to his stock of arithmetical beliefs – that is how he likes to do arithmetic. Compare:

[7] Feferman's introduction in Gödel (1990, pp. 282–287) seems to attribute to Gödel a more complex line of argument, and he briefly suggests using Jeroslow's variant version of the Second Theorem to throw light on Gödel's thinking. Michael Potter follows Feferman and attempts to develop this interpretation at greater length in Potter (2000, Section 10.3); however Richard Zach has shown that Potter's treatment is technically flawed (Zach, 2005). Which makes me hesitate to go beyond the simple reading of Gödel's remarks that I've given. And those remarks are surely interesting enough even on the simple reading.

Kurt also follows where his eyes take him – in the sense that, if he takes himself to see that ψ, he (generally) comes to believe that ψ.

Kurt, let's suppose, now gets reflective about his fact-gathering. He comes to realize that, for lots of instances, when he seems to see that ψ, then it *is* the case that ψ (at least, that's how he supposes the world to go – and what other vantage point can he take?). And he finds no reason not to continue generally trusting his eyes, meaning that he is prepared more generally to endorse instances of the schema: when he seems to see that ψ, then ψ.

Similarly, Kurt reflects that when he can derive φ in PA, then it is the case that φ (at least, that's how he takes the arithmetical world to go). When it isn't the case that φ, he can't ever derive φ in PA: and so he finds no reason not to endorse his own continuing confidence in PA. So he is disposed to accept the conditional: when PA entails φ, then φ. Suppose that, for whatever reason, Kurt is especially interested in Π_1 arithmetical claims. Then, in particular, Kurt is disposed to accept instances of that conditional when φ is Π_1.

Next, Kurt hits one day on the idea of systematically introducing a code-numbering scheme which associates wffs and sequences of wffs with numbers (he discovers how to arithmetize syntax). And he realizes that the relation $Prf(m, n)$ which holds when m codes for a PA derivation of the wff with number n can be captured in PA by a long and complicated wff $\mathsf{Prf}(\mathsf{x}, \mathsf{y})$; and hence Kurt comes to see that there is a wff $\mathsf{Prov}(\mathsf{x}) =_{\text{def}} \exists \mathsf{v}\, \mathsf{Prf}(\mathsf{v}, \mathsf{x})$ which expresses provability in PA. Since for any Π_1 sentence φ, Kurt will happily accept that *if* PA *entails* φ *then* φ, he will now equally happily accept its arithmetical correlate $\mathsf{Prov}(\ulcorner\varphi\urcorner) \rightarrow \varphi$. Kurt, however, although bold in his enthusiasm for PA, is fundamentally a cautious man, as befits someone with his concern for rigour: so he *doesn't* suppose that all these instances of the reflection schema are themselves already proved by PA (which is a good thing, since we know from Löb's Theorem that PA *doesn't* prove such an instance unless it also proves φ itself). In other words, Kurt cheerfully allows that what follows from his original theory PA alone might be less than what follows from PA plus an arithmetization of his new thought that PA is reliable for Π_1 sentences: he never supposed that PA had to be the last word about the truths of L_A.

So Kurt is now in the position of accepting the axioms of PAΠ, i.e. the axioms of PA plus all instances of its reflection schema $\Box\varphi \rightarrow \varphi$ where φ is a Π_1 sentence. Hence, Kurt has come to accept a theory in which he can produce a derivation of PA's canonical Gödel sentence as in our proof of Theorem 27.5 (if and when he gets round to spotting the construction). And such a derivation will, for him, count as a *proof* of G, as it is done within the framework of PAΠ which he now accepts.

We could even, if we like, give the story a fanciful dramatic twist if we imagine Kurt proceeding as follows. We could imagine him *first* proving PA's canonical Gödel sentence in PAΠ, before he slaps his forehead in surprise as he sees that he also has a simple argument that PA doesn't prove G – thus showing that his earlier caution in not assuming that PA was the last word about arithmetic was

well placed. (Though perhaps, like Isaacson, he still thinks that PA is the last word on purely arithmetical reasoning about numbers, for he recognizes that his new assumption which takes him beyond PA comes from reflections not just on numbers but on a formal arithmetical theory.)

So we seem to have got to the following point: Kurt *could* come to accept G *without* going via a semantic theory for the language of PA. He needs, it seems, no explicit thoughts about models for PA, or about PA's consistency. Rather, he just notes his own confidence in PA's arithmetical reliability, and endorses it.

You might protest: 'Hold on! Kurt wouldn't accept that $0 = 1$ should it turn out that PA is inconsistent and proves $0 = S0$, would he? So it is only reasonable for him to be confident in instances of PA's reflection schema if he has a reason for thinking he isn't in for a really nasty surprise, i.e. if he *already* has a reason for thinking that PA is at least consistent. In other words, to follow through the suggested line of reasoning to the conclusion that G is true, Kurt after all does have to engage with some sort of argument – e.g. the specification of a model for PA – that could justify a belief in PA's consistency.'

But while this protest at first sight looks compelling, on reflection it is based on what is – to say the least – a deeply problematic epistemological assumption. To be sure, were it turn out that PA 'proves' $0 = S0$, Kurt would abandon his confidence in PA. But why should we assume that it follows from *that* that Kurt needs some guarantee that PA won't deliver a nasty surprise if he is to be reasonable in moving from accepting PA (as he does) to accepting instances of its reflection schema? Compare: should I suddenly start seeing a crazy world of flying pigs and talking donkeys, I'll stop believing my eyes. But why should we assume that it follows from that that I need some guarantee in advance that things won't go crazy and that my eyes are (generally) reliable when I endorse the thought that in fact what my eyes tell me is the case (generally) *is* the case? In the world as it actually is, it is reasonable for me to reflectively endorse the presumption that my eyes are reliable, in the absence of countervailing considerations ('reasonable' in the sense that it is quite appropriate default behaviour for a responsible cognitive agent): it is similarly reasonable for Kurt to put his continued trust in PA in the absence of nasty surprises.

Of course, the route we've described which ends up with Kurt believing G, starting from an acceptance of PA's reflection principle, isn't available to get Kurt to endorse a canonical Gödel sentence for a theory he *doesn't* accept (like the theory $Q^\dagger$ which we defined in Section 18.3): in *that* sort of case, Kurt has to have sophisticated ideas about truth-in-a-model, or some such. The point we are making here is that this isn't how it *has* to be across the board.[8]

[8] We have been touching on themes discussed by Ketland (1999, 2005) and Tennant (2002, 2005).

28 Interlude: About the Second Theorem

The title of Gödel's great paper is 'On formally undecidable propositions of *Principia Mathematica* and related systems I'. And as we noted in Section 18.5, his First Incompleteness Theorem does indeed undermine *Principia*'s logicist ambitions. But logicism wasn't really Gödel's main target. For, by 1931, much of the steam had already gone out of the logicist project. Instead, the dominant project for showing that classical infinitary mathematics is in good order was Hilbert's Programme, which we have already mentioned a few times. This provided the real impetus for Gödel's early work; it is time we filled out just a bit more of the story.

However, this book certainly isn't the place for a detailed treatment of the changing ideas of Hilbert and his followers as their ideas developed pre- and post-Gödel; nor is it the place for an extended discussion of the later fate of Hilbertian ideas.[1] So our necessarily brief remarks will do no more than sketch the logical geography of some broadly Hilbertian territory: those with more of a bent for the history of logic can be left to fight over e.g. the question of Hilbert's precise path through the landscape.

Another topic we'll take up in this Interlude is the vexed one of the impact of the incompleteness theorems, and in particular the Second Theorem, on the issue of mechanism: do Gödelian results show that minds cannot be machines?

28.1 'Real' vs 'ideal' mathematics

What does it take to grasp the truths of arithmetic – what does it take, for example, to grasp that every number has a successor? Kant famously thought that it involves the exercise of *'intuition'*, whatever exactly that is: it requires some cognitive source that goes beyond what is given by analytic reflection on the logical concepts which we deploy in thinking about any subject matter at all. Frege equally famously disagreed. His fundamental claim is that

> Pure thought (irrespective of any content given by the senses or even by an intuition a priori) can, solely from the content that results from its own constitution, bring forth judgements that at first sight appear to be possible only on the basis of some intuition. (Frege, 1972, §23)

But sadly, as we've already noted, his logicist attempt to derive all of arithmetic from logic-plus-definitions became tangled in contradiction (Section 10.8). And

[1] For more, see Zach (2003), much expanded to Zach (2006).

Russell and Whitehead's attempt to develop a paradox-free foundation for arithmetic is also highly problematic when regarded as an attempt to vindicate the logicist project: for example, how can *Principia*'s Axiom of Infinity genuinely be counted as a logical axiom?

However, even if we don't regard it as a purely logical system, it is still highly interesting that *Principia* gives us a unified framework in which we can regiment not just arithmetic but a great deal of mathematics. Though standard Zermelo-Fraenkel set theory plus the Axiom of Choice turns out in fact to be a much neater framework to use.

But hold on! If neither *Principia*'s system nor ZFC can now be advertised as belonging to pure logic, how can we choose which to use? Which is '*true*'? Both? Or should we perhaps use neither, but adopt some other kind of set theory?

Faced with puzzling questions like that, it is rather tempting to suppose that they are intractable because misguided. Perhaps we just shouldn't think of infinitary set theories and other mathematical exotica as being in the business of truth or falsity. Rather, to put it in Hilbertian terms, we should divide mathematics into a core of uncontentious *real* mathematics and a superstructure of *ideal* mathematics. 'Real' mathematics is to be taken at face value (it expresses contentful propositions, or is 'contentual' for short); and the propositions of real mathematics are straightforwardly true or false. So, for example, very elementary arithmetic is 'real' – it really is the case that two plus three is five, and it really isn't the case that two plus three is six. Perhaps more generally, Π_1-statements of arithmetic are 'real'. By contrast, 'ideal' mathematics shouldn't be thought of as having representational content, and its sentences aren't strictly speaking true or false at all – perhaps some parts of ideal mathematics are instrumentally useful tools, helping us to establish 'real' truths, while other parts are just intellectual *jeux d'esprit*. Of course, there will be straightforward truths about what follows from what in a particular ideal mathematical game: the claim, though, is that the statements made *within* the game are like statements within a fiction, not themselves straightforwardly either true or false. So, for example, perhaps the more ambitious parts of arithmetic, and certainly the wilder reaches of infinitary set theory, are to be treated as 'ideal' in this sense.

In pursuing this idea, Hilbert himself was inclined to take a *very* restricted view of 'real' mathematics. In part, that was a strategic ploy: the idea is to count as real mathematics only some uncontroversial core of arithmetic which even the most stringent critic of infinitary mathematics is happy to accept. But Hilbert was mainly swayed by Kantian ideas. He thinks of the most certain core of arithmetic as grounded in 'intuition', in our apprehension of finite sequences and of the results of manipulating these. And he thinks that 'intuition' is enough to give us knowledge of the results of simple arithmetic operations on particular numbers, and also of Π_1 generalizations about these operations. But we can't pursue here the vexed question of how far 'intuition' can take us,[2] and so we'll

[2]For discussion, see for example Parsons (1980, 1998), Tait (1981, 2002).

put on hold the question of where exactly we might want to place the dividing line between core 'real' mathematics and 'ideal' mathematics. We'll soon see, however, that given his wider purposes Hilbert needs to be right on one thing: 'real' mathematics needs to include at least the arithmetic of Π_1 statements.

28.2 A quick aside: Gödel's caution

Hilbertians, then, thought that the status of most mathematics is to be sharply distinguished from that of some small central core of 'real', true, elementary arithmetic. Most mathematics is merely 'ideal': we can talk about what demonstrably follows from what within the game, but shouldn't talk of the statements made in the game as being true. At least in part for that reason, as Gödel put it in a letter, at the time

> . . . a concept of objective mathematical truth as opposed to demonstrability was viewed with greatest suspicion and widely rejected as meaningless. (Gödel, 2003a, p. 10, fn. c)

Which probably explains why – as we've remarked before – Gödel in 1931 very cautiously downplayed the version of his incompleteness theorems that depended on the assumption that the theories being proved incomplete are *sound* (i.e. have *true* theorems), and instead put all the weight on the purely syntactic assumptions of consistency or ω-consistency. For more on this historical point, see Feferman (1984).

28.3 Relating the real and the ideal

Let's now ask: what relations might hold between Hilbert's two domains of mathematics? For brevity, we'll use the following symbols:

i. I is a particular ideal theory ('ideal' need not mean 'purely ideal': an ideal theory can extend a contentual one).

ii. C is our best correct theory of contentual real mathematics. Being correct, C is consistent, and all its deductive consequences are true. (We can leave it open whether C is a tidily axiomatizable theory.)

So here are four relations that an ideal theory I might have to real mathematics. In each case we are generalizing only over *real*, contentual, propositions φ:

1. If $I \vdash \varphi$, then $C \vdash \varphi$.
2. If $I \vdash \varphi$, then φ is true.
3. If $I \vdash \varphi$, then $C \nvdash \neg\varphi$.
4. If $I \vdash \varphi$ and C decides φ, then $C \vdash \varphi$.

In the first case, we'll say that I is *real-conservative*: the ideal theory can only prove real propositions that we can already prove in our core contentual theory. In the second case, I is *real-sound*: the ideal theory can only prove *true* contentual propositions. In the third case, we'll say that I is *real-consistent* – i.e. I can't prove anything inconsistent with theorems of core real mathematics. Finally, in the fourth case, I is *weakly-conservative* – i.e. I agrees with C at least on the contentual real propositions that C can decide one way or the other.

Trivially, all four conditions require I to be consistent. And the relations between the conditions are now easily seen to be as follows:

$$(1) \rightarrow (2) \rightarrow (3) \leftrightarrow (4)$$

Proof If I is real-conservative it must be real-sound (since all C's entailments are true). But not vice-versa. For C might well be sound but not complete.

If I is real-sound it is evidently real-consistent. But again not necessarily vice-versa. For suppose that C isn't negation complete, and doesn't decide the contentual proposition φ. If we merely know that I is real-consistent, then discovering that $I \vdash \varphi$ by itself leaves it open that we might equally have $I' \vdash \neg\varphi$ for some other ideal theory I': and I and I' can't both be real-sound.

Finally, if I is real-consistent it is weakly-conservative. Assume that $C \nvdash \neg\varphi$ if $I \vdash \varphi$; then if C decides φ, that means $C \vdash \varphi$. While for the reverse implication, suppose that I is weakly-conservative. Suppose too that $I \vdash \varphi$ but also $C \vdash \neg\varphi$ for some contentual φ. Then C decides φ; so by weak conservativeness $C \vdash \varphi$ making C inconsistent, contrary to the assumption that C is correct. Hence, if $I \vdash \varphi$, then $C \nvdash \neg\varphi$, and so I is real-consistent. ⊠

28.4 Proving real-soundness?

(a) Suppose that you *are* attracted by this plan of distinguishing a core of 'real', straightforwardly true, mathematics from the great superstructure of 'ideal' mathematics. Then you'll want to know which bits of ideal mathematics are safe to use, i.e. don't lead you to false real beliefs, i.e. are real-sound. And, to the extent that you can ratify theories as real-sound, you will then have vindicated the practice of infinitary mathematics. Even though, *sotto voce*, you'll say to yourself that only real mathematics is genuinely true, you can plunge in and play the ratified games of ideal mathematics with a clear conscience, knowing that they can't lead you astray.

Which suggests a Hilbertian programme, very different from the logicist programme of trying to derive everything, including standard infinitary mathematics, from logic-plus-definitions. The new programme is: *seek to defend those parts of ideal mathematics we take to be useful in extending our contentual knowledge by showing them to be real-sound.*[3]

[3]Those bits of ideal mathematics which are hermetically sealed games, with no contentual implications, can be left to look after themselves!

Now, we have already seen in Sections 27.5 and 27.7 that there are Gödelian limitations on the provability of real-soundness. But rather than jump straight to pressing that point, let's proceed more slowly, pretending for a moment that we are still in a state of pre-Gödelian innocence.

(b) Even prescinding from Gödelian considerations, it might seem that the Hilbertian project is doomed from the very outset. For how can we possibly show that an ideal theory I has true contentual consequences without assuming that the relevant axioms used in deriving these consequences are true and hence are contentual and hence are not really ideal after all?

But in fact, there *are* a couple of routes by which we could in principle lever ourselves up from a belief that I is *consistent* and has some correct real content to the conclusion that I is overall *real-sound*. We rely on a couple of easy theorems:

> **Theorem 28.1** *If I is consistent and extends C, and C is negation-complete, then I is real-sound.*

Proof Suppose φ is a contentual proposition and $I \vdash \varphi$ but φ is false. Since C by hypothesis is correct, $C \nvdash \varphi$. Since C is negation-complete, that implies $C \vdash \neg\varphi$. But by the definition of 'extends', if $C \vdash \neg\varphi$, $I \vdash \neg\varphi$. So I is inconsistent contrary to hypothesis. So if $I \vdash \varphi$ then φ is true. ⊠

> **Theorem 28.2** *If I is consistent and extends* Q*, and if contentual mathematics extends no further than Π_1 propositions of arithmetic, then I is real-sound.*

Proof This is just half of Theorem 9.4 in our new jargon, since Π_1-soundness implies real-soundness if contentual mathematics extends no further than the Π_1 propositions of arithmetic. ⊠

Given these mini-theorems, we can immediately discern the shape of two possible lines of argument for defending the use of an ideal theory in establishing contentual truths. We won't worry too much about whether either is the historical Hilbert's own mature programme for defending the useful branches of ideal mathematics: but they are surely both Hilbertian arguments in a broad sense.

H1. First option. We start by characterizing real contentual mathematics (perhaps quite generously). We then establish (i) that there is in fact a negation-complete theory C for this real mathematics, and also establish (ii) that our favourite useful theories of ideal mathematics like ZF set theory both extend C and are consistent. Then by Theorem 28.1 we'll be entitled to believe the contentual implications of ZF (or whatever) because we'll have a warrant for the claim that they are already entailed by C and so are true.[4] (The ideal theories can't prove anything that wasn't already provable in C: but going via the ideal theories might offer much shorter and/or much clearer proofs.)

[4]Here is Hilbert, seemingly endorsing a general argument from consistency to real-soundness:

H2. Second option. We restrict real mathematics to Π_1 claims of arithmetic – as it seems did Hilbert himself. Establish that favourite theories in ideal mathematics like ZF set theory both extend Q and are consistent. Then by Theorem 28.2 we'll be entitled to believe the contentual implications of ZF (even though, in this case, we *won't* always already be able to deduce them in a restricted contentual theory).

(c) Now, both these lines of argument require us to establish the consistency of axiomatized theories in ideal maths in order to prove that they are 'safe'.[5] But what does 'establish' mean here? Given that the overall project is to find a respectable place for ideal mathematics (in particular, infinitistic mathematics) as an instrumentally useful supplement to real mathematics, 'establishing' can't involve appeal to the very same infinitary ideas which we are trying to legitimate. So consistency will have to be established by appeal to nothing more exotic than the kosher 'safe' reasoning already involved in contentual mathematics.

But how can we get common-or-garden contentual mathematics to touch directly on questions of the consistency of formal axiomatized theories? By the arithmetization of syntax, of course. But recall, a consistency claim about an ideal theory I is canonically expressible by a Π_1 sentence Con_I. So *if consistency proofs are to be within reach of contentual mathematics, then contentual mathematics must – as we said – be able to cope at least with some* Π_1 *claims of arithmetic* (so presumably must include at least Q).

28.5 The impact of Gödel

(a) Now trouble! First, if our contentual mathematics is to be regimented sufficiently well for the question of establishing that it is negation-complete to arise, then it will need to be a properly axiomatized theory C. But no properly axiomatized contentual theory C including Q can be negation-complete, by Gödel's First Theorem. So the First Theorem is *already* enough to sabotage the first Hilbertian programme (H1).

> For there is a condition, a single but absolutely necessary one, to which the use of the method of ideal elements is subject, and that is the proof of consistency; for, extension by the addition of ideals is legitimate only if no contradiction is thereby brought about in the old, narrower domain, that is, if the relations that result for the old objects whenever the ideal objects are eliminated are valid in the old domain. (Hilbert, 1925, p. 383)

Hilbert doesn't there fully explain his thought, nor does he explicitly assert the negation-completeness of real mathematics which is needed for the obvious argument to go through. However, his pupil and collaborator Paul Bernays does:

> In the case of a finitistic proposition ... the determination of its irrefutability is equivalent to determination of its truth. (Bernays, 1930, p. 259)

It seems quite a reasonable bet that Hilbert agreed. See Raatikainen (2003).

[5] In fact, Gödel discovered his incompleteness theorems while trying to prove the consistency of classical analysis.

That leaves the second programme (H2) still in the hunt, as *that* doesn't require any assumptions about completeness. However we now know from Gödel's Second Theorem that no modest formal arithmetic can establish the consistency of a fancy ideal theory. So the second programme (H2) must fail too as the desired 'contentual' consistency proofs for branches of ideal mathematics won't be forthcoming.

Or at least, those are the obvious claims about the impact of Gödel's Theorems on the general Hilbertian project of trying to establish the real-soundness of ideal theories by giving consistency proofs. Is there any wriggle room left?

(b) Michael Detlefsen (1986) has mounted a rear-guard defence of Hilbert that, in part, plays with the thought that we should consider Rosserized ideal theories where we ensure that the proofs are consistency-minded – i.e. a sequence of wffs counts as a proof of φ only if there is no 'earlier' proof of $\neg\varphi$, etc. But we have already noted that, while we can trivially prove such theories to be consistent, the idea of making play with consistency-minded proofs is beset with difficulties (Section 27.2). And in any case, the idea can't in fact be used to rescue a version of our second Hilbertian programme, for the following reason.

(H2) depends on Theorem 28.2, which tells us that if I is consistent and extends Q, then I is real-sound – assuming real mathematics goes no further than Π_1 truths. Now, for normal theories, it can of course be easy to show that I extends Q (just show that I proves Q's axioms). If, however, I_R is a Rosserized theory, then – while we can trivially see that it can't Rosser-prove contradictions – *we can't in general effectively decide whether it extends* Q. For suppose $\mathsf{Q} \vdash \varphi$. Then even if I_R proves Q's axioms, it doesn't follow that there is a consistency-minded proof of φ, because for all we know there could be an I_R proof of $\neg\varphi$ which is shorter than the shortest Q proof of φ. So if we Rosserize our theories – or fiddle with proof predicates in similar ways – then we can't make use of (H2) to show that our ideal theories are real-sound.

(c) So, is there any *other* route to establishing real-soundness for ideal theories, using only relatively modest arithmetic reasoning? Well, if we continue to suppose that real mathematics must be able to do Π_1 arithmetic, so real-soundness embraces Π_1-soundness, then we know that there can't be. That is, of course, what is shown by Gödel's 'best and most general version of the unprovability of consistency in the same system' (see Section 27.7). Modest arithmetic reasoning can't even prove the Π_1-soundness of modest arithmetics, let alone the Π_1-soundness of more fancy theories.

So the 'best version' would seem to mark the end of the story. But not quite. For remember, in Section 24.7 we very briefly outlined Gentzen's consistency proof for PA. To be sure, that uses reasoning which goes beyond the narrowly finitary; but it might perhaps still be defended as belonging to 'safe' real mathematics. Exploring such prospects for limited consistency proofs using safe-but-not-strictly-finitary methods would, however, take us far too far afield.

And let's not complicate matters. Whatever the options for descendants of

Hilbert's programme, the headline news remains this: the Hilbertian project in anything very close to its original form is sunk by Gödel.[6]

28.6 Minds and computers

(a) We now turn to another matter on which Gödel's Theorems impact. Consider the following first-shot argument:

> Call the set of mathematical sentences which I accept, or at least could derive from what I accept, my *mathematical output* O. And consider the hypothesis that there is some kind of computing machine which can in principle list off my mathematical output – i.e., it can effectively enumerate (the Gödel numbers for the sentences in) O. Then O is effectively enumerable, and by Theorem 19.4 it follows that there is a p.r. axiomatized theory M whose theorems are exactly my mathematical output. Since I accept the axioms of Q plus, indeed, some induction, O is at least as strong as $\mathsf{I}\Sigma_1$, and so M is p.r. adequate. So I can now go on to prove that M can't prove its canonical Gödel sentence G_M. But in going through that proof, I will come to establish by mathematical reasoning that G_M is true. Hence M does not, after all, entail *all* my mathematical output. Contradiction. So no computer can effectively generate my mathematical output. Even if we just concentrate on my mathematical abilities and potential mathematical output, I can't be emulated by a mere computer!

This style of argument is often presented as leading to the conclusion that I can't be emulated by a 'Turing machine' in particular (see Section 2.2, (c)). But note that if there is any force to the sketched argument, it will apply to computing devices more generally – which is why we can discuss it here, before we get round to explaining the special notion of a Turing machine.

One immediate problem with this kind of argument, of course, is the unclarity of the idea of my 'mathematical output'. Is it to contain just what I could derive, given my limited cognitive abilities, my limited life, etc? In that case, my output is finite; and then quite trivially we know the argument goes wrong – because for any finite set of Gödel numbers, there will trivially be a computer that can enumerate it, given enough memory storage (just build the finite list into its data store). So to get an argument going here, we'll have to radically idealize my mathematical abilities, for a start by allowing me to follow a proof of arbitrary length and complexity. But what other idealizations are allowed? That's unclear.

[6]From the start, Gödel himself left open the possibility that there could be a Hilbertian project which relied on a richer kind of consistency proof: see Gödel (1931, p. 195). For some relevant later investigations, see Gödel (1958). And for general discussion see also Giaquinto (2002, Part V, Ch. 2), as well as Zach (2003, 2006).

However, even setting aside that point, the argument is in bad shape, ultimately for a very simple reason already prefigured in Section 18.4 (a). Grant that I can in general establish that *if* the axiomatized theory T is consistent and contains enough arithmetic, then G_T is true. But of course, assuming T has enough induction, T itself can also prove $\mathsf{Con}_T \to \mathsf{G}_T$ (see Theorem 25.4). Apply that to the particular case where T is the theory M that generates my mathematical output. Then even if I can establish that if M is consistent, then G_M is true, M can prove that too (since it will contain $\mathsf{I}\Sigma_1$): there is no difference yet between my output and M's.

Now, if I could now go on to establish that M *is* consistent, then that would indeed distinguish me from M, because I can then establish M's canonical Gödel sentence is true, and M can't. But we've so far been given no reason to suppose that I *can* show that M is consistent, even if idealized. In fact, we've been given no reason to suppose that I'll even know what theory M is.[7]

(b) Can we improve the first-shot argument? To make progress, we need to be entitled to the thought that the relevant M is consistent. Of we course, we *hope* that our mathematical output O is consistent, and so correspondingly we will *hope* that M is consistent. But wishful thinking isn't an argument.[8] However, perhaps we can get somewhere if we think of our mathematical output not as defined in terms of what we *accept* and can derive from what we accept, but in terms of what we *know* because we can prove it true (since every sentence we can prove true must be consistent with every other sentence we can prove true).

So let's consider the following argument, which is essentially due to Paul Benacerraf (1967). Let's now use K for the set of mathematical truths that are knowable by me (idealizing, take it to be the deductive closure of what I

[7] The locus classicus for this point is Putnam (1960).

[8] John Lucas has urged that the hypothesis that there is a machine that emulates me is only worth considering given that the mechanist makes a consistency assumption:

> Putnam's objection fails on account of the dialectical nature of the Gödelian argument. . . . [T]here is a claim being seriously maintained by the mechanist that the mind can be represented by some machine. Before wasting time on the mechanist's claim, it is reasonable to ask him some questions about his machine to see whether his seriously maintained claim has serious backing. It is reasonable to ask him not only what the specification of the machine is, but whether it is consistent. Unless it is consistent, the claim will not get off the ground. If it is warranted to be consistent, then that gives the mind the premiss it needs. The consistency of the machine is established not by the mathematical ability of the mind but on the word of the mechanist. (Lucas, 1996, p. 113)

But the issue isn't whether the *machine* which is supposed to be emulating my mathematical output is consistent – it churns away, effectively enumerating a set of Gödel numbers, and it can be in as good order as any other computing machine. The question is whether the sentences which the Gödel numbers encode form a consistent set. Even if they don't, that set could still be my idealized mathematical output: for example, perhaps the ZFC set theory I accept is inconsistent, but the shortest proof of inconsistency is far too long for anyone actually to grasp – which is why, as real-world unidealized mathematicians, we haven't noticed the contradictions which lurk over the horizon, far down the road.

can prove true by mathematical argument). And again suppose that there is a computer which emulates me in the sense that it effectively enumerates K. By the same argument as before that entails the assumption

1. There is a p.r. axiomatized theory N such that, for all φ, $\varphi \in K \leftrightarrow N \vdash \overline{\varphi}$.

where $\overline{\varphi}$ is N's formal counterpart for the informal claim φ. And now let's *also* assume that one of the broadly mathematical things that I know is that the theory N indeed generates my output of mathematical knowledge. Equivalently,

2. 'for all φ, $\varphi \in K \leftrightarrow N \vdash \overline{\varphi}$' $\in K$

We can then continue as follows:

3. By hypothesis, everything in K is true: so K is consistent – or Con_K for short. And since that's a mini-proof of Con_K, then 'Con_K' $\in K$.

4. Since Con_K, and a sentence is in K if and only if its formal counterpart is provable in N, then N is consistent too, i.e. Con_N for short. So, since we've just proved *that*, '$Con_K \wedge$ (for all φ, $\varphi \in K \leftrightarrow N \vdash \overline{\varphi}$) $\rightarrow Con_N$' $\in K$.

5. Since K by hypothesis is deductively closed, (2), (3) and (4) imply 'Con_N' $\in K$.

6. So by (1) again, $N \vdash \mathsf{Con}_N$, where Con_N formally expresses the consistency of N.

7. But since I know quite a bit of arithmetic to be true (more than $\mathsf{I}\Sigma_1$), enough arithmetic must be built into N for the Second Theorem to apply. Hence $N \nvdash \mathsf{Con}_N$. Contradiction!

8. So either assumption (1) or assumption (2) has to be false.

In other words, either my (idealized) mathematical knowledge isn't capturable in a theory describing the potential output of a computing machine or, if it is, I don't know which theory, and hence which machine, does the trick. Which is quite neat,[9] but also perhaps fairly unexciting. After all, it isn't exactly easy to tell which bits of my putative mathematical knowledge really *are* knowledge (perhaps ZFC is inconsistent after all!): so why on earth suppose that I'd have the god-like ability to correctly spot the computer program that actually gets things right in selecting out (the deductive closure of) what I truly know? So we can and should cheerfully embrace the second limb of the disjunctive conclusion.

(c) In his Gibbs lecture, Gödel himself considered the impact of the Second Theorem on issues about minds and machines (Gödel, 1951). Like Benacerraf,

[9]If it is indeed legitimate to assume that it is the same formal consistency statement that is involved at steps (6) and (7): but let's grant that it is.

he reaches a disjunctive conclusion.[10] Gödel starts by remarking that the Second Theorem

> ... makes it impossible that someone should set up a certain well-defined system of axioms and rules and consistently make the following assertion: All of these axioms and rules I perceive (with mathematical certitude) to be correct, and moreover I believe that they contain all of mathematics. If someone makes such a statement he contradicts himself. For if he perceives the axioms under consideration to be correct, he also perceives (with the same certainty) that they are consistent. Hence he has a mathematical insight not derivable from his axioms. (Gödel, 1951, p. 309)

So this is the now familiar thought that we can keep on extending sound theories to get new ones by adding their consistency sentences as new axioms (see Section 25.4). But how far can we follow through this process? Either we say 'infinitely far': at least in principle, idealizing away from limitations on memory and time and so forth, we can keep on going for ever, grasping ever more extensive systems of arithmetic as evidently correct, but never completing the task. Or we say 'the human mind (even if we bolt on more memory and abstract from time constraints etc.) loses its grip at some point: and then there are further truths that remain for ever beyond the reach of proof'. Gödel puts the alternatives this way:

> Either mathematics is incompletable in this sense, that its evident axioms can never be comprised in a finite rule, that is to say, the human mind (even within the realm of pure mathematics) infinitely surpasses the powers of any finite machine, or else there exist absolutely [unprovable Π_1 sentences] ... where the epithet 'absolutely' means that they would be [unprovable], not just within some particular axiomatic system, but by *any* mathematical proof the human mind can conceive.[11]

Now, there are questions which can be raised about this argument:[12] but perhaps the principal point to make is that, even if the argument works, its disjunctive conclusion is again anodyne. Anyone of naturalist inclinations will be happy enough to agree that there are limits on the possibilities of human mathematical cognition, even if we abstract from constraints of memory and time. Gödel himself was famously not naturalistically inclined and, according to Hao Wang, he was inclined to reject the second disjunct Wang (1974, pp. 324–326): but for once there seems no evident good reason to follow Gödel here.

[10] An outline of Gödel's position was reported by Wang (1974, pp. 324–326); but the lecture wasn't published until 1995.

[11] From Gödel (1951, p. 310). I've reversed the order of the passages either side of the lacuna. Also, Gödel talks of 'unsolvable diophantine equations' rather than, equivalently, about unprovable Π_1 sentences: see Gödel (1951, p. 157) for an explanation of the connection.

[12] See, for example, the discussion in Feferman (2006).

(d) So are there *other* arguments that lead from thoughts about Gödelian incompleteness to more substantial, and non-disjunctive, conclusions?

Well, there is indeed a battery of attempts to find such arguments, starting with a much-cited paper by John Lucas (1961) and latterly continuing in books by Roger Penrose (1989, 1994). We certainly haven't space to follow all the twists and turns in the debates here. But it is fair to say that these more intricate anti-mechanist arguments based on the incompleteness theorems have so far produced *very* little conviction. If you want to explore further, Stewart Shapiro's rich (1998) makes an excellent place to start.

28.7 The rest of this book: another road-map

We have now proved Gödel's First Incompleteness Theorem and outlined a proof of his Second Theorem.

And it is worth stressing that the ingredients used in our discussions so far have really been *extremely* modest. We introduced the ideas of expressing and capturing properties and functions in a formal theory of arithmetic, the idea of a primitive recursive function, and the idea of coding up claims about relations between wffs into claims about relations between their code-numbers. We showed that some key numerical relations coding proof relations for sensible theories are p.r., and hence can be expressed and indeed captured in any theory that includes Q. Then, in the last dozen chapters, we have worked Gödelian wonders with these very limited ingredients. We haven't need to deploy any of the more sophisticated tools from the logician's bag of tricks. Note, in particular, that in proving our formal theorems, we *haven't* yet had to call on a general theory of computable functions or (equivalently) on a general theory of effectively decidable properties and relations.

Compare our incompleteness theorems in Chapters 5 and 6. Theorem 5.7 says: if T is a sound axiomatized theory whose language is sufficiently expressive, then T cannot be negation complete. Theorem 6.2 says: a consistent, sufficiently strong, axiomatized formal theory of arithmetic cannot be negation complete. A 'sufficiently expressive language', remember, is one which could express at least every effectively decidable two-place numerical relation, and a 'sufficiently strong theory' is one which can capture at least all effectively decidable numerical properties. So those informal theorems *do* deploy the notion of effective decidability. And to get the introductory part of the book and its informal completeness theorem to fit together nicely with our later official Gödelian proofs, we'll therefore need to give a formal treatment of decidability.

So that's our main task in the remaining chapters. Of course, we are not aiming here for a very extensive coverage of the general theory of computability (that would require a book in itself); we'll just be concentrating on a handful of central topics which are most immediately relevant to developing our understanding of incompleteness theorems. So, in more detail, here's what lies ahead:

1. We first extend the idea of a primitive recursive function in a natural way, and define a wider class of intuitively computable functions, the *μ-recursive* functions. We give an initial argument for *Church's Thesis* that these μ-recursive functions comprise *all* total numerical functions which are effectively computable. (Chapter 29)

2. We already know that Q, and hence PA, can capture all the p.r. functions: we next show that they can capture all the μ-recursive functions. The fact that Q and PA are recursively adequate immediately entails that neither theory is decidable – and it isn't mechanically decidable either what's a theorem of first-order logic. We can also quickly derive the formal counterpart of the informal syntactic incompleteness theorem of Chapter 6. (Chapter 30)

3. We then turn to introduce another way of defining a class of intuitively computable functions, the *Turing-computable* functions: *Turing's Thesis* is that these are exactly the effectively computable functions. We go on to outline a proof that the Turing-computable (total) functions are in fact just the μ-recursive functions again. (Chapters 31, 32)

4. Next we prove another key limitative result (i.e. a result, like Gödel's, about what *can't* be done). There can't be a Turing machine which solves the *halting problem*: there is no general effective way of telling in advance whether an arbitrary machine with program Π ever halts when it is run from input n. We show that the unsolvability of the halting problem gives us another proof that it isn't mechanically decidable what's a theorem of first-order logic, and it also entails Gödelian incompleteness again. (Chapter 33)

5. The fact that two independent ways of trying to characterize the class of computable functions coincide supports what we can now call the *Church-Turing Thesis*, which underlies the links we need to make e.g. between formal results about what a Turing machine can decide and results about what is effectively decidable in the intuitive sense. We finish the book by discussing the Church–Turing Thesis further, and consider its status. (Chapters 34, 35)

29 μ-Recursive functions

This chapter introduces the notion of a μ-recursive function – which is a very natural extension of the idea of a primitive recursive function. Plausibly, the effectively computable functions are exactly the μ-recursive functions (and likewise, the effectively decidable properties are those with μ-recursive characteristic functions).

29.1 Minimization and μ-recursive functions

The primitive recursive functions are the functions which can be defined using *composition* and *primitive recursion*, starting from the successor, zero, and identity functions. These functions are computable. But they are not the only computable functions defined over the natural numbers (see Section 11.5 for the neat diagonal argument which proves the point). So the natural question to ask is: what other ways of defining new functions from old can we throw into the mix in order to get a broader class of computable numerical functions (hopefully, to get *all* of them)?

As explained in Section 11.4, p.r. functions can be calculated using *bounded* loops (as we enter each 'for' loop, we state in advance how many iterations are required). But as Section 3.6 already reminds us, we also count *unbounded search* procedures – implemented by 'do until' loops – as computational. So, the obvious first way of extending the class of p.r. functions is to allow functions to be defined by means of some sort of 'do until' procedure. We'll explain how to do this in four steps.

(a) Here's a simple example of a 'do until' loop in action. Suppose that G is a decidable numerical relation. And suppose that for every x there is a y such that Gxy. Then, given a number x, we can find a G-related number y by the brute-force algorithmic method of running through the numbers y from zero up and deciding in each case whether Gxy, until we get a positive result.

Suppose that G's characteristic function is the function g (so Gxy holds just when $g(x, y) = 0$). Then the algorithm can be presented like this:

1. $y := 0$
2. Do until $g(x, y) = 0$
3. $\quad y := y + 1$
4. Loop
5. $f(x) := y$

Here, we set y initially to take the value 0. We then enter a loop. At each iteration, we do a computation to decide whether the Gxy holds, i.e. whether $g(x, y) = 0$. If it does, we exit the loop and put $f(x)$ equal to the current value of y; otherwise we increment y by one and do the next test. By hypothesis, we do eventually hit a value of y such that $g(x, y) = 0$: the program is bound to terminate. So this 'do until' routine calculates the number $f(x)$ which is *the least y such that $g(x, y) = 0$*, i.e. the least number to which x is G-related. This algorithm, then, gives us a way of effectively calculating the values of a new total function f, given the function g.

(b) Now let's generalize this idea. Let $\vec{x}$ stand in for n variables. Then we'll say that

> The $(n+1)$-place function $g(\vec{x}, y)$ is *regular* iff it is a total function and for all values of $\vec{x}$, there is a y such that $g(\vec{x}, y) = 0$.

Suppose g is a regular computable function. Then the following routine will effectively compute another function $f(\vec{x})$:

1. $y := 0$
2. Do until $g(\vec{x}, y) = 0$
3. $\quad y := y + 1$
4. Loop
5. $f(\vec{x}) := y$

By hypothesis g is a total computable function, so at the k-th loop checking whether $g(\vec{x}, k) = 0$ is a mechanical business which always delivers a verdict. By hypothesis again, g is regular, so the looping procedure eventually terminates for each $\vec{x}$. Hence f is a total computable function, defined for all arguments $\vec{x}$.

Here's another definition:

> Suppose $g(\vec{x}, y)$ is an $(n + 1)$-place regular function. Let $f(\vec{x})$ be the n-place function which, for each $\vec{x}$, takes as its value the least y such that $g(\vec{x}, y) = 0$. Then we say that f is defined by *regular minimization* from g.

Then what we've just shown is that *if f is defined from the regular computable function g by regular minimization, then f is a total computable function too,* with values of f effectively computable using a 'do until' routine.[1]

(c) Now some notation. Recall, in Section 11.8 we introduced the symbolism 'μy' to abbreviate 'the least y such that ...'. So now, when f is defined from g by regular minimization, we can write:

[1] If we drop the requirement that g is regular, the 'do until' procedure may sometimes, or even always, fail to produce an output: it will at most compute a *partial* function $f(\vec{x})$ which is defined perhaps for only some or even for no values. The theory of partial computable functions is a very important rounding out of the general theory of computable functions. But we don't need to tangle with it in this book. All the functions we'll be talking about are *total* functions, as we'll keep emphasizing from time to time.

$$f(\vec{x}) = \mu y[g(\vec{x}, y) = 0]$$

(The square brackets here are strictly speaking unnecessary, but are fairly standard, and greatly aid readability.) When $g(\vec{x}, y)$ is the characteristic function of the relation $G(\vec{x}, y)$, we will occasionally write, equivalently,

$$f(\vec{x}) = \mu y[G(\vec{x}, y)]$$

Compare, then, the operation of bounded minimization which we met in Section 11.8: we are now concerned with a species of *unbounded* minimization.

(d) Summarizing so far: we said that we can expect to expand the class of computable functions beyond the p.r. ones by considering functions that are computed using a 'do until' search procedure. We've just seen that when we define a function by regular minimization, this in effect specifies that its value is to be computed by just such a search procedure. Which suggests that a third mode of definition to throw into the mix for defining computable functions, alongside composition and primitive recursion, is definition by regular minimization.

With that motivation, let's say:

> The *μ-recursive* functions are those that can be defined from the initial functions by a chain of definitions by composition, primitive recursion and/or regular minimization.[2]

Or putting it more carefully, we can say

1. The initial functions S, Z, and I_i^k are μ-recursive;
2. if f can be defined from the μ-recursive functions g and h by composition, then f is μ-recursive;
3. if f can be defined from the μ-recursive functions g and h by recursion, then f is μ-recursive;
4. if g is a *regular* μ-recursive function, and f can be defined from g by regular minimization, then f is μ-recursive;
5. nothing else is a μ-recursive function.

Since regular minimization yields total functions, the μ-recursive functions are always total computable functions. Trivially, all p.r. functions also count as μ-recursive functions.

29.2 Another definition of μ-recursiveness

This little section is just to forestall a query which might already have occurred to you!

[2] Many, perhaps most, writers nowadays use plain 'recursive' instead of 'μ-recursive'. But the terminology hereabouts can be confusingly variable. It will do no harm, then, to stick to our explicit label.

A 'for' loop – i.e. a programming structure which instructs us to iterate some process as a counter increments from 0 to n – can of course be recast as a 'do until' loop which tells us to iterate the same process while incrementing the counter *until its value equals* n. So it looks as if definitions by primitive recursion, which call 'for' loops, could be subsumed under definitions by minimization, which call 'do until' loops. Hence you might well suspect that clause (3) in our definition is redundant. And you'd be *almost* though not quite right. By a theorem of Kleene's (1936c), you can indeed drop (3) *if* you add addition, multiplication and the characteristic function of the less-than relation to the list of initial functions. And some books define recursive functions this way; see e.g. Shoenfield (1967, p. 109).

Still, I for one don't find this approach nearly as natural or illuminating, so let's stick to the more conventional mode of presentation given in the previous section.

29.3 The Ackermann-Péter function

Since μ-recursive functions can be defined using unbounded searches and p.r. functions can't, we'd expect there to be μ-recursive functions which aren't primitive recursive. But can we give some examples?

Well, the computable-but-not-p.r. function $d(n)$ that we constructed by the diagonalization trick in Section 11.5 is in fact an example. But it isn't immediately obvious *why* the diagonal function is μ-recursive. So in this section and the next we'll look at another example, which is both more tractable and also mathematically more natural. The basic idea is due to Wilhelm Ackermann (1928).

Let's begin with a simple observation. Any p.r. function f, recall, can be specified by a chain of definitions in terms of primitive recursion and composition leading back to initial functions. This definition won't be unique: there will always be various ways of defining f (for a start, by throwing in unnecessary detours). But take the shortest definitional chain – or, if there are ties for first place, take one of the shortest. Now, the length of this shortest definitional chain for f will evidently put a limit on how fast $f(n)$ can grow as n grows. That's because it puts a limit on how complicated the computation can be – in particular, the number of loops-within-loops-within-loops that we have to play with. And so it limits the number of times we ultimately get to apply the successor function, depending on the initial input argument n. A similar point applies to two-place functions, etc.

That's a bit abstract, but the point is easily seen if we consider the two-place functions f_1, i.e. *sum* (repeated applications of the successor function), f_2, i.e. *product* (repeated sums), f_3, i.e. *exponentiation* (repeated products). These functions have increasingly long full definitional chains; and the full programs for computing them involve 'for' loops nested increasingly deeply. And as their respective arguments grow, the value of f_1 of course grows comparatively slowly,

f_2 grows faster, f_3 faster still.

This sequence of functions can obviously be continued. Next comes f_4, the *super-exponential*, defined by repeated exponentiation:

$$x \Uparrow 0 = x$$
$$x \Uparrow Sy = x^{x \Uparrow y}$$

Thus, for example, $3 \Uparrow 4$ is $3^{3^{3^3}}$ with a 'tower' of 4 exponents. Similarly, we can define f_5 (super-duper-exponentiation, i.e. repeated super-exponentiation), f_6 (repeated super-duper-exponentiation), and so on. The full chain of definitions for each f_k gets longer and longer as k increases – and the values of the respective functions grow faster and faster as their arguments are increased.[3]

But now consider the function $a(x) = f_x(x, x)$. The value of $a(x)$ grows *explosively*, running away ever faster as n increases. Indeed, take any given one-place p.r. function: this has a maximum rate of growth determined by the length of its definition, i.e. a rate of growth comparable to some $f_n(x, x)$ in our hierarchy; but $a(x)$ eventually grows faster than any particular $f_n(x, x)$. Hence $a(x)$ isn't primitive recursive. Yet it is evidently computable.

This idea of Ackermann's is *very* neat, and is worth pausing over and developing a bit. So consider again the recursive definitions of our functions f_1 to f_4 (look again at Section 11.1, and at our definition of '$\Uparrow$' above). We can rewrite the second, recursion, clauses in each of those definitions as follows:

$$\begin{aligned} f_1(y, Sz) &= Sf_1(y, z) \\ &= f_0(y, f_1(y, z)) \text{ – if we cunningly define } f_0(y, z) = Sz \\ f_2(y, Sz) &= f_1(y, f_2(y, z)) \\ f_3(y, Sz) &= f_2(y, f_3(y, z)) \\ f_4(y, Sz) &= f_3(y, f_4(y, z)) \end{aligned}$$

There's a pattern here! So now suppose we put

$$f(x, y, z) =_{\text{def}} f_x(y, z)$$

Then the value of f gets fixed via a *double* recursion:

$$f(Sx, y, Sz) = f(x, y, f(Sx, y, z))$$

However, nothing very exciting happens to the second variable, 'y'. So we'll now let it just drop out of the picture, and relabel the remaining variable to get a variant on the Ackermann's construction due to Rósza Péter (1935). Consider, then, the function p governed by the clause

$$p(Sx, Sy) = p(x, p(Sx, y))$$

Of course, this single clause doesn't yet fully define p – it doesn't tell us, e.g., the value of $p(0, 0)$. So we need somehow to round out the definition, e.g. to yield the following three equations:

[3]The claim, of course, isn't that longer definitions always *entail* faster growth, only that our examples show how longer definitions *permit* faster growth.

$$p(0, y) = Sy$$
$$p(Sx, 0) = p(x, S0)$$
$$p(Sx, Sy) = p(x, p(Sx, y))$$

To see how these equations work together to determine the value of p for given arguments, work through the following calculation:

$$\begin{aligned}
p(2,1) &= p(1, p(2, 0)) \\
&= p(1, p(1, 1)) \\
&= p(1, p(0, p(1, 0))) \\
&= p(1, p(0, p(0, 1))) \\
&= p(1, p(0, 2)) \\
&= p(1, 3) \\
&= p(0, p(1, 2)) \\
&= p(0, p(0, p(1, 1))) \\
&= p(0, p(0, p(0, p(1, 0)))) \\
&= p(0, p(0, p(0, p(0, 1)))) \\
&= p(0, p(0, p(0, 2))) \\
&= p(0, p(0, 3)) \\
&= p(0, 4) \\
&= 5
\end{aligned}$$

To evaluate the function, the recipe is as follows. At each step look at the innermost occurrence of p, and apply whichever of the definitional clauses pertains – it's trivial to check that only one can. Keep on going until at last you reach something of the form $p(0, m)$ and then apply the first clause one last time and halt.

A little reflection will convince you that this procedure *does* always terminate (hint: look at the pattern of numbers in vertical columns, and also along diagonals). And note that our informal recipe in effect involves a *do until* procedure. So – given everything we've said – it shouldn't be a surprise to hear that we have the following:

> **Theorem 29.1** *The Ackermann-Péter function is μ-recursive but not primitive recursive.*

We'll outline a proof of the *first* half of this double-barrelled claim in the next section. We won't pause so long over the *second* half, however, as we've already indicated the main proof idea: here it is again ...

Sketch of a proof sketch: $p(x, y)$ is not p.r. Given p's origin in the sequence of functions sum, product, exponential, ..., it is evident that the functions $p(0, y)$, $p(1, y)$, $p(2, y)$, ... grow ever faster as y increases. And so if $y > 0$, and $n > m$, $p(n, y) > p(m, y)$.

We can also fairly easily confirm that, for any primitive recursive function $f(y)$ – whose rate of growth is capped by the length of its shortest p.r. definition

– there is a corresponding n such that $p(n, y)$ grows faster.[4] In other words, after some threshold d, then for all $y > d$, $f(y) < p(n, y)$. Hence p can't be primitive recursive.

For suppose p is p.r., and then put $f(y) = p(y, y)$. Evidently, f will be p.r. too. Hence there is some n and some d such that, if $y > d$, then $f(y) < p(n, y)$. So now chose some y' such that $y' > d$, $y' > n$: then $f(y') < p(n, y') < p(y', y') = f(y')$. Contradiction! ⊠

29.4 The Ackermann-Péter function is μ-recursive

We'll take longer over proving the first half of Theorem 29.1; that's because the proof introduces a neat strategy which we'll later make use of a number of times. So it's worth getting the hang of the argument.

Proof sketch: $p(x, y)$ is μ-recursive Our outline proof has three stages, the first two just setting up some coding and defining a couple of coding functions.

(i) *Introducing more coding* Consider the successive terms in our calculation of the value of $p(2, 1)$. We can introduce code numbers representing these terms by a simple, two-step, procedure:

α. Transform each term like $p(1, p(0, p(1, 0)))$ into a corresponding sequence of numbers like $\langle 1, 0, 1, 0\rangle$ by the simple expedient of deleting the brackets and occurrences of the function-symbol 'p'. (We can uniquely recover terms from such sequences in the obvious way.)

β. Code the resulting sequence $\langle 1, 0, 1, 0\rangle$ by Gödel numbering, e.g. by using powers of primes. So we put e.g.

$$\langle l, m, n, o\rangle \Rightarrow 2^{l+1} \cdot 3^{m+1} \cdot 5^{n+1} \cdot 7^{o+1}$$

(where we need the $+1$ in the exponents to handle the zeros).

Hence we can think of our computation of $p(2, 1)$ as generating in turn the α-sequences

$$\langle 2, 1\rangle, \langle 1, 2, 0\rangle, \langle 1, 1, 1\rangle, \langle 1, 0, 1, 0\rangle, \ldots$$

Then we code up each such α-sequence; so the successive steps in the calculation of $p(2, 1)$ will respectively receive the β-code numbers

72, 540, 900, 2100, ...

(ii) *Coding/decoding functions* So far, that's just routine coding. Now we put it to work by defining a couple of coding/decoding functions as follows:

[4] The details get rather tiresome however, and we won't give them here. See e.g. Cohen (1987, §3.6).

i. $c(x, y, z)$ is the β-code of the α-sequence corresponding to the output of the z-th step (counting from zero) in the calculation of $p(x, y)$ if the calculation hasn't yet halted by step z; and otherwise $c(x, y, z) = 0$. So, for example, $c(2, 1, 0) = 72$, $c(2, 1, 3) = 2100$, and $c(2, 1, 20) = 0$.

ii. $fr(x)$ = one less than the exponent of 2 in the prime factorization of x. Hence, if n is a β-code for a sequence of numbers, $fr(n)$ recovers the first member of the sequence.

(iii) *Facts about our coding functions* Next, here is a number of claims about our coding functions, which together establish the desired result that the Ackermann-Péter function is μ-recursive.

1. The coding function $c(x, y, z)$ is primitive recursive. That's because the evaluation of c for arguments l, m, n evidently involves a step-by-step numerical computation tracking the first n steps in the calculation of $p(l, m)$, using the very simple rules that take us from one step to the next. No step-to-step move involves flying off on an open-ended search. So 'for' loops will suffice to construct an algorithm for the computation of c. And such an algorithm will always determine a p.r. function.

   That's quick-and-dirty: making the argument watertight is pretty tedious though not difficult. There's nothing to be learnt from spelling out the details here: so we won't.

2. The calculation of the function p for given arguments x, y eventually halts at the z-th step for some z, and then $c(x, y, Sz) = 0$. Hence c is regular.

3. $\mu z[c(x, y, Sz) = 0]$ is therefore the step-number of the final step in the calculation which delivers the value of $p(x, y)$. Since c is regular, it follows that $\mu z[c(x, y, Sz) = 0]$ defines a μ-recursive function.

4. Hence $c(x, y, \mu z[c(x, y, Sz) = 0])$ gives the code number of the final value of $p(x, y)$. Since this compounds a p.r. (hence μ-recursive) function with a μ-recursive one, it is also μ-recursive.

5. Hence, decoding, $p(x, y) = fr(c(x, y, \mu z[c(x, y, Sz) = 0]))$.

6. But the function fr is primitive recursive (in fact $fr(x) =_{\text{def}} exp(x, 0) \dot{-} 1$, where exp is as introduced in Section 11.8).

7. Thus $fr(c(x, y, \mu z[c(x, y, Sz) = 0]))$ is the composition of a p.r. function and a μ-recursive function. Hence it, i.e. $p(x, y)$, is μ-recursive. ⊠

29.5 Introducing Church's Thesis

(a) That was a delightful argument! And the strategy it deploys is evidently a powerful one. For example – although we won't give the details here – the

computable-but-not-p.r. diagonal function $d(n)$ from Section 11.5 can similarly be shown to be μ-recursive by a broadly similar proof.[5]

And now generalizing, we might reasonably expect to be able to code up the step-by-little-step moves in *any* well-defined calculation by a primitive recursive coding function like c (primitive recursive because, when broken down to minimal steps, there again won't be any flying off on open-ended searches). If the output of the calculation is defined for every input, then exactly the same style of argument will be available to show that the mapping from input to output must be a μ-recursive function.

This line of thought – which has its roots in Church (1936b) – very strongly encourages the conjecture that in fact *all* effectively computable total functions will turn out to be μ-recursive.

Reflections on modern programming languages point in the same direction. For when things are reduced to basics, we see that the main programming structures available in such languages are (in effect) 'for' loops and 'do until' loops, which correspond to definitions by primitive recursion and minimization. Hence – given that our modern general-purpose programming languages have so far proved sufficient for specifying algorithms to generate any computable function we care to construct – it doesn't seem a very big leap to conjecture that every algorithmically computable total function should be definable in terms of composition (corresponding to the chaining of program modules), primitive recursion, and minimization.

In sum, such considerations certainly give a very high initial plausibility to what's called

[5]See Péter's classic (1951), with a revised edition translated as her (1967). The key idea is to use a double recursion again to define a function $\varphi(m, n)$ such that, for a given m, $\varphi(m, n) = f_m(n)$, where running through the f_i gives us our effective enumeration of the p.r. functions. And since φ is definable by a double recursion it can be shown to be μ-recursive by the same kind of argument which showed that the Ackermann-Péter function is μ-recursive. Hence $d(n) = \varphi(n, n) + 1$ is μ-recursive too.

Just for enthusiasts: it is perhaps worth footnoting that it would be quite wrong to take away from our discussion so far the impression that μ-recursive-but-not-p.r. functions must all suffer from explosive growth. Péter gives a beautiful counter-example. Take our enumeration f_i of p.r. functions, and now consider the functions $g_i(n) =_{\text{def}} sg(f_i(n))$, where sg is as defined in Section 11.8 (i.e. $sg(k) = 0$ for $k = 0$, and $sg(k) = 1$ otherwise). Evidently, running through the g_i gives us an effective enumeration – with many repetitions – of all the p.r. functions which only take the values 0 and 1. Now consider the μ-recursive function $\psi(n) = \overline{sg}(\varphi(n, n)) = |1 - sg(\varphi(n, n))|$ (where φ is as above). This function too only takes the values 0 and 1; but it can't be primitive recursive. For suppose otherwise. Then for some k, $\psi(n) = g_k(n) = sg(\varphi(k, n))$. So we'd have

$$sg(\varphi(k, n)) = \psi(n) = |1 - sg(\varphi(n, n))|$$

and hence

$$sg(\varphi(k, k)) = \psi(k) = |1 - sg(\varphi(k, k))|$$

Which is impossible. Therefore there are μ-recursive-but-not-p.r. functions which only ever take the values 0 and 1, and hence do not suffer value explosion. However, while *values* of such functions can remain tame, *lengths of computations* don't, as we'll see in Section 33.6, fn. 5. There remains a sense, then, in which μ-recursive-but-not-p.r. functions are *wild*.

> *Church's Thesis* The total numerical functions that are effectively computable by some algorithmic routine are just the μ-recursive functions.[6]

And certainly all the evidence supports this Thesis. For a start, no one has ever been able to define an intuitively computable numerical total function which *isn't* μ-recursive.

We'll be saying a lot more about all this later. Pending further discussion, however, we'll for the moment just *assume* that Church's Thesis is true. Given this assumption, the class of μ-recursive functions is indeed of very special interest as it just *is* the class of effectively computable numerical functions.

29.6 Why can't we diagonalize out?

You might find that last claim very puzzling (in fact, if you've been following, perhaps you *ought* to find it puzzling!). For don't we already have all the materials to hand for a knock-down argument against Church's Thesis? Back in Section 11.5, we proved that not every computable function is primitive recursive by the trick of 'diagonalizing out'. That is to say, we used a diagonal construction which took us from a list of all the p.r. functions to a further computable function which *isn't* on the list. Why shouldn't we now just use the same trick again to diagonalize out of the class of μ-recursive functions?

Well, the argument would have to go:

> Take an effective enumeration of the μ-recursive functions, f_0, f_1, f_2, ..., and define the diagonal function $d(n) = f_n(n) + 1$. Then d differs from each f_j (at least for the argument j). But d is computable (since to evaluate it for argument n, you just set a computer to enumerate the f_j until it reaches the n-th one, and then by hypothesis the value of $f_n(n) + 1$ is computable). So d is computable but not μ-recursive.

But this argument fails, and it is very important to see why. The crucial point is that we are not entitled to its initial assumption. While the p.r. functions are effectively enumerable, *we can't assume that there is an effective enumeration of the μ-recursive functions.*

What makes the difference? Well, remind yourself of the informal argument (in Section 11.5) that shows that we can mechanically list off the recipes for the p.r. functions. If we now try to run a parallel argument for the claim that the μ-recursive functions are effectively enumerable, things go just fine at the outset:

[6] The reason for the label will emerge in Chapter 34. Compare *Turing's Thesis* which we very briefly introduced in Section 2.2: that says that the (total) numerical functions that are effectively computable by some algorithmic routine are just those functions that are computable by a Turing machine (which is a computer following a very simple-minded type of program: for more explanation, see Chapter 31). It turns out that our two Theses are equivalent, because the μ-recursive functions are exactly the Turing computable ones, as we'll show in Chapter 32.

> Every μ-recursive function has a 'recipe' in which it is defined by primitive recursion or composition or regular minimization from other functions which are defined by recursion or composition or regular minimization from other functions which are defined ultimately in terms of some primitive starter functions. So choose some standard formal specification language for representing these recipes. Then we can effectively generate 'in alphabetical order' all possible strings of symbols from this language ...

But at this point the parallel argument breaks down, since we *can't* continue

> ... and as we go along, we can mechanically select the strings that obey the rules for being a recipe for a μ-recursive function.

That's because, in order to determine mechanically whether a series of definitions obey the rules for being the recipe for a μ-recursive function, we'd need an effective way of determining whether each application of the minimization operator is an application to a *regular* function. We'd need a way of effectively determining whether e.g. a p.r. function $g(x, y)$ is such that for each x there is a y such that $g(x, y) = 0$. And there is in general no effective way of doing that.

It's worth adding another observation. For note that we *know* that there can't be an effective enumeration of the *effectively* computable total functions f_0, f_1, f_2, ... For if there were one, we could define $d(n) = f_n(n)+1$ which would then evidently be an effectively computable function not on the list. Contradiction.

Since we know the effectively computable total functions are not effectively enumerable, we certainly can't just *assume* that there is an effective enumeration of the μ-recursive functions. To use the 'diagonalizing out' argument against Church's Thesis, we'd *already* need some independent reason for thinking the enumeration can be done. There isn't any.

In sum: Church's Thesis that the μ-recursive functions are *all* the (total, numerical) computable functions lives to fight another day.[7]

29.7 Using Church's Thesis

Church's Thesis, to repeat, is a biconditional: a total numerical function is μ-recursive if and only if it is effectively computable in the intuitive sense. Half the Thesis is quite unexciting – if a function is μ-recursive, then it is certainly computable. It is the other half which is the interesting claim, the half which says that if a total numerical function is *not* μ-recursive then it is *not* computable in the intuitive sense.

Over the coming chapters, we'll repeatedly be appealing to Church's Thesis, but in two quite different ways which we need to distinguish very clearly. Let's call these the *interpretive* and the *labour-saving* uses respectively.

[7] Let's stress again: it is important that we have been talking throughout about *total* computable functions.

The interpretive use relies on the Thesis to pass from technical claims about what is or isn't μ-recursive to claims about what is or isn't effectively computable. Here, then, the Thesis is being used to justify an informal gloss on our technical results. And if we are in general to interpret formal results about μ-recursiveness as telling us about computability in the intuitive sense, then necessarily we have to appeal to the Thesis.

The labour-saving use relies on the Thesis to pass in the opposite direction, from informal claims about what is or isn't computable in the intuitive sense to formal claims about μ-recursiveness. So in particular, it allows us to jump from a quick-and-dirty informal proof that something is effectively computable to conclude that a corresponding function is μ-recursive. This kind of fast-track argument for some technical claim is fine, given that Church's Thesis is entirely secure. However, any formal result which can be established this way by appeal to the Thesis *must* be provable directly, the hard way (otherwise we would have located a disconnect between the informal notion of computability and its claimed formal equivalent, contradicting the Thesis). Hence this second sort of use of the Thesis is labour-saving but is always inessential.

For clarity's sake, we'll adopt the following convention in the rest of this book. When we simply say 'by Church's Thesis ...', or 'given Church's Thesis ...', etc. we'll always be appealing to Church's Thesis in the first way, to make a connection between a formal claim and a claim about computability in the intuitive sense. When we occasionally make use of Church's Thesis in the second way, to support a technical claim that some function is μ-recursive, then we'll explicitly signal what we are doing: we'll say 'by a labour-saving appeal to Church's Thesis' or some such.

30 Undecidability and incompleteness

Theorem 13.6 tells us that Q can capture all p.r. functions. Our next theorem shows that Q can in fact capture all μ-recursive functions. With a bit of help from Church's Thesis, our new stronger theorem enables us very quickly to prove two new Big Results: first, any nice theory is undecidable; and second, theoremhood in first-order logic is undecidable too.

The old Theorem 13.6 is, of course, the key result which underlies incompleteness theorems like Theorem 17.2 (if T is nice and ω-consistent, then T is incomplete). Our new theorem correspondingly underlies some easy variations on that earlier incompleteness theorem and its relatives. We'll also prove a formal counterpart to the informal theorem of Chapter 6.

30.1 Q is recursively adequate

Recall that we said that a theory is p.r. adequate if it captures each p.r. function as a function (Section 12.4). Let's likewise say that

> A theory is *recursively adequate* iff it captures each μ-recursive function as a function.

We showed that Q is p.r. adequate in Chapter 13. Overall that took some ingenuity; but given the work we've already done, it is now very easy to go on to establish

Theorem 30.1 Q *is recursively adequate.*

Proof Theorem 13.3 tells us that Q can capture any Σ_1 function as a function. To establish that Q is recursively adequate, it therefore suffices to show that recursive functions are Σ_1 (i.e. are expressible by Σ_1 wffs).

And to show *that*, it suffices to prove that (a) the initial functions are Σ_1, and that if the functions g and h are Σ_1, so are the functions derived from these by repeated steps of (b) substitution, (c) primitive recursion and (d) regular minimization. But we know that (a) to (c) are true from the proof of Theorem 13.5. So we just need to check that the extra condition (d) also obtains.

Take the case where the one-place function f is defined by regular minimization from the two-place function g, so $f(x) = \mu y[g(x, y) = 0]$. And suppose that $g(x, y)$ is expressed by the strictly Σ_1 predicate $\mathsf{G}(\mathsf{x}, \mathsf{y}, \mathsf{z})$. Then f is evidently expressed by

$$\mathsf{F}(\mathsf{x}, \mathsf{y}) =_{\text{def}} \mathsf{G}(\mathsf{x}, \mathsf{y}, 0) \wedge (\forall \mathsf{u} \leq \mathsf{y})(\mathsf{u} \neq \mathsf{y} \rightarrow \exists \mathsf{z}(\mathsf{G}(\mathsf{x}, \mathsf{u}, \mathsf{z}) \wedge \mathsf{z} \neq 0))$$

Now recall the quantifier-shifting trick that we used in Section 13.6. So, first step, we see that the following expresses the same function:

$$\mathsf{F}'(\mathsf{x},\mathsf{y}) =_{\text{def}} \mathsf{G}(\mathsf{x},\mathsf{y},0) \wedge \exists \mathsf{w}(\forall \mathsf{u} \leq \mathsf{y})(\mathsf{u} \neq \mathsf{y} \rightarrow (\exists \mathsf{z} \leq \mathsf{w})(\mathsf{G}(\mathsf{x},\mathsf{u},\mathsf{z}) \wedge \mathsf{z} \neq 0))$$

Now use the same trick again to take any initial unbounded existential quantifiers in G and drag them forward past the bounded quantifiers, leaving behind their bounded 'shadows'. Then we'll get another wff F'' that is strictly Σ_1, but still expresses the same function. So f is Σ_1.

The generalization to many-place functions is immediate: so we are done. ⊠

Recall the idea of a 'sufficiently strong' theory, i.e. the idea of a theory that captures all intuitively decidable one-place properties of numbers, i.e. the idea of a theory that captures all effectively computable two-place characteristic functions. By Church's Thesis (used in interpretive mode), a theory which is recursively adequate is sufficiently strong. So Theorem 30.1 (at long last) redeems our promise to vindicate the intuitive notion of a sufficiently strong theory. Such a theory need be no richer than our old friend, the decidedly tame theory Q!

Finally, given that Q captures all μ-recursive functions as functions, so does any nice theory, including PA of course. And such theories will also capture all μ-recursive properties, i.e. properties with μ-recursive characteristic functions – compare the little connecting result in Section 12.4 (b).

30.2 Nice theories can *only* capture recursive functions

The result that Q, and hence PA, can capture all recursive functions is the one which we will be repeatedly using. But there's also a converse result:

> **Theorem 30.2** *If T is nice, any total function which can be captured in T is μ-recursive.*

A proof using Church's Thesis Take the monadic case (it will be obvious how to generalize). Suppose the total function $f(m)$ can be captured as a function in T. Then, by definition, there is a two-place open wff $\varphi(\mathsf{x},\mathsf{y})$ which is such that if $f(m) = n$, $T \vdash \varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$; and if $f(m) \neq n$, $T \vdash \neg\varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$, so – since T is consistent by assumption – $T \nvdash \varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$. Trivially, then, the value of $f(m)$ is the least number n (indeed, the only number n) such that $T \vdash \varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$.

So, for given m, we can effectively compute the value of $f(m)$ using an open-ended search. Start effectively enumerating the T-theorems, and keep on going until one of the form $\varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$ turns up. The value of $f(m)$ is the resulting value of n. (We know we can effectively enumerate the T-theorems by Theorem 3.5; and because f is total, the search for an instance of $\varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$ always terminates).

Since there is therefore an algorithm for computing the value of f, it follows by Church's Thesis (used in labour-saving mode) that f is μ-recursive. ⊠

One quick comment. If Q and PA can capture all the μ-recursive functions but no more, it follows that Q and PA capture exactly the same functions. Is

that surprising? After all, PA can prove a *lot* more than Q! Why can't it prove more function-capturing wffs? Well, it *can* prove lots more function-capturing wffs: but they will at most give us more complex ways of capturing the same old functions for which Σ_1 wffs suffice.

30.3 Some more definitions

Let's pause for some reminders about old bits of jargon, interlaced with definitions for some fairly self-explanatory bits of new jargon.

i. First recall from Sections 2.2 and 11.6 the informal idea of a decidable property, i.e. a property P whose characteristic function c_P is computable.

We'll now say that a numerical property P is *recursively decidable* iff its characteristic function c_P is μ-recursive – i.e. iff there is a μ-recursive function which, given input n, delivers a 0/1, yes/no, verdict on whether n is P. The definition obviously extends in a natural way to cover recursively decidable numerical relations.

We will say that a non-numerical property is recursively decidable iff some acceptable Gödel-style coding associates the property with a recursively decidable numerical property (and again similarly for relations).[1]

We'll also say that a *set* is recursively decidable iff the property of being a member of the set is recursively decidable.

Church's Thesis implies, of course, that the intuitively decidable properties/relations/sets are just the recursively decidable ones.

ii. Here's some equally standard alternative jargon. A (numerical) *decision problem* takes the form: to tell of any arbitrary number n whether it is P (or to tell of an arbitrary pair of numbers m, n, whether m has relation R to n, etc.). Such a decision problem is said to be *recursively solvable* iff there is a μ-recursive function which, given any input n, delivers a verdict on whether n is P (likewise for relations). To say that the decision problem for P is recursively solvable is therefore just to say that the property P is recursively decidable.

iii. When we very first characterized the idea of a formal theory, we said that for a properly constructed theory it must be decidable what's a wff, what's an axiom, and what's a well-constructed logical derivation. If we use the Gödel-numbering trick, then the requirement becomes that the properties of numbering a wff, axiom or correct derivation must be decidable. Previously, we introduced the notion of a p.r. axiomatized theory, which is one for which those three numbering properties are primitive recursive (are decidable by p.r. functions). So now let's correspondingly say

[1] By the argument of Section 15.1 (b), whether a non-numerical property is recursively decidable does not depend on our choice of Gödel coding. (Exercise: check this!)

that a theory is *recursively axiomatized* when the same three properties meet the less stringent requirement of being recursively decidable.

By Church's Thesis, the formalized theories in the intuitive sense are exactly the recursively axiomatized ones.

iv. We said in Section 3.4 (ii) that a theory is decidable iff the property of being a theorem of that theory is decidable. So we'll correspondingly define a theory to be *recursively decidable* iff it is recursively decidable what's a theorem.

v. Finally, recall from Section 2.4 that we said that a set is effectively enumerable iff it is either empty or there is an effectively computable (total) function which enumerates it. Putting that more carefully, Σ is effectively enumerable iff it is either empty or there is a surjective[2] effectively computable (total) function $f\colon \mathbb{N} \to \Sigma$. In other words, Σ is either empty or the range of an effectively computable function.

Similarly, then, we'll now say that a numerical set $\Sigma \subseteq \mathbb{N}$ is *recursively enumerable* iff it is either empty or there is a μ-recursive function which enumerates it – i.e. iff Σ is either empty or it is the range of a μ-recursive function. And by extension, a non-numerical set Σ is recursively enumerable iff, under an acceptable system of Gödel coding, the set Σ' which contains the code numbers of members of Σ is recursively enumerable.

It is standard to abbreviate 'recursively enumerable' just by 'r.e.'. And Church's thesis tells us, of course, that a set is r.e. if and only if it is effectively enumerable in the intuitive sense.

30.4 Q and PA are undecidable

Now down to new business again! We start with a Big Result which is immediate, given what we've already shown:

Theorem 30.3 *If T is a nice theory, it isn't recursively decidable.*

Proof To say that T is recursively decidable is to say that the property $Prov_T$ of numbering a T-theorem is μ-recursive (i.e. has a μ-recursive characteristic function). Suppose, then, that $Prov_T$ *is* μ-recursive. Then $Prov_T$ would be capturable in T since, being nice, T contains Q and so is recursively adequate by Theorem 30.1. However, Theorem 21.3 tells us that no open wff in a nice theory T can capture $Prov_T$. Contradiction. ⊠

If Q is consistent, it is nice. Hence if Q is consistent, it isn't recursively decidable. Likewise, if PA is consistent, it isn't recursively decidable. And assuming Church's Thesis as well, Q *and* PA *are therefore undecidable in the intuitive sense* (compare Section 6.2).

[2] In case you've forgotten: a function $f\colon \mathbb{N} \to \Sigma$ is *surjective* iff the range of f is the whole of Σ, i.e. for every $y \in \Sigma$ there is some $x \in \mathbb{N}$ such that $f(x) = y$. See Section 2.3.

30.5 The *Entscheidungsproblem*

Q is such a *very* simple theory we might quite reasonably have hoped that there *would* be some mechanical way of telling which wffs are and which aren't its theorems. But we now know that there isn't. And in a sense, you can blame the underlying first-order logic. For, assuming Q's consistency, we get *Church's Theorem* as an immediate corollary of Q's recursive undecidability:

> **Theorem 30.4** *The property of being a theorem of first-order logic is recursively undecidable.*

Proof Suppose first-order theoremhood is recursively decidable. In other words, suppose that the property of numbering a logical theorem is μ-recursive. Let $\hat{\mathsf{Q}}$ be the conjunction of the seven non-logical axioms of Q, and let φ be any sentence of L_A. By our supposition, there is therefore a μ-recursive function which decides whether $(\hat{\mathsf{Q}} \rightarrow \varphi)$ is a logical theorem. But $(\hat{\mathsf{Q}} \rightarrow \varphi)$ is a logical theorem just if φ is a Q-theorem. So our supposition implies that there is a μ-recursive function which decides what's a theorem of Q. But we've just shown there can be no such function, given Q's consistency. So the supposition must be false.[3] ⊠

The so-called *Entscheidungsproblem*, the problem of coming up with an effective method for deciding of an arbitrary sentence of first-order logic whether it is valid or not, was famously posed by Hilbert and Ackermann in §11 of their *Grundzüge der theoretischen Logik* (1928), the first recognizably modern logic textbook. Our new Big Result says that there is in fact no recursive function that takes (the g.n. of) a sentence and returns a 0/1, yes/no, verdict on theoremhood. So given Church's Thesis, *it follows that the problem of deciding first-order theoremhood is not just recursively unsolvable but unsolvable, period.*[4]

30.6 Incompleteness theorems again

Next, let's very quickly revisit some earlier Gödelian incompleteness results (which we established without any reference to a general theory of computable functions) in order to link them to the versions which are found in many modern treatments (where computability and the theory of μ-recursive functions are often discussed first).

[3] And now we see the very particular interest in finding a recursively adequate (and hence undecidable) arithmetic like Q which has only a *finite* number of axioms. We couldn't have similarly shown the undecidability of first-order logic by invoking the undecidability of PA, for example, because we couldn't start 'Let P be the conjunction of the axioms of PA'. PA has an infinite number of axioms: remember all those instances of the induction schema! And indeed – although we can't prove it here – PA *essentially* has an infinite number of axioms: there is no equivalent theory with the same theorems which only has a finite number of axioms (Kaye, 1991, p. 132).

[4] This was first shown, independently, by Church (1936a, b) and Turing (1936), by different routes: see also Section 33.3.

The formal analogue of the intuitive notion of a formalized theory is the idea of a *recursively* axiomatized theory. However, when we proved Gödel's results about the limitations of formal arithmetics, we didn't discuss recursively axiomatized theories of arithmetic but – rather more narrowly – *primitively recursively* axiomatized theories. Does that matter?

Well, as we've already noted in Section 19.1, focusing on p.r. axiomatized theories isn't really a restriction either in practice or in principle. That's because recursively but not p.r. axiomatized theories will in practice be very peculiar beasts (for example: to set things up so we have to check that a putative axiom *is* an axiom by an open-ended 'do until' search is just not a natural way of constructing an axiomatized theory). And a version of Craig's Theorem tells us that, in any case, an axiomatized theory T can in principle always be given a p.r. re-axiomatization (i.e. there's a p.r. axiomatized theory T' which will have the same theorems as T, and so T' will be negation-complete iff T is). So the First Theorem which proves the incompleteness of consistent, rich enough, p.r. axiomatized theories thereby proves the incompleteness of *any* consistent, rich enough, formalized theory. We could therefore very well leave matters as they were, happily concentrating on p.r. axiomatized theories, without missing out on anything very important.

But for the record, let's now note that the key theorems we proved e.g. for *nice* theories (i.e. consistent p.r. axiomatized theories which include Q) can be extended – without appealing to Craig's Theorem – to apply directly to the wider class of *nice*$^+$ theories, i.e. consistent *recursively* axiomatized theories which include Q. For just one example, corresponding to Theorem 17.2 we have

> **Theorem 17.2*** *If T is a nice$^+$ theory then there is an L_A-sentence φ of Goldbach type such that $T \nvdash \varphi$ and (if T is also ω-consistent) $T \nvdash \neg\varphi$.*

Proof sketch We essentially replicate the arguments for the original theorem. The crucial point of difference will be that, if we are only told that T is recursively axiomatized (as opposed to p.r. axiomatized), then we can only show that the corresponding relation Gdl_T is μ-recursive (as opposed to being primitive recursive). That's because Gdl_T is defined in terms of Prf_T which is in turn defined in terms of the likes of $Axiom_T$, and we are now only given that such basic properties are μ-recursive. But no matter. T includes the axioms of Q, so T is recursively adequate by Theorem 30.1, so T can still capture Gdl_T. By the proof of Theorem 30.1, a Σ_1 wff suffices. And the rest of the argument runs on exactly the same lines as before. ⊠

In the same way, we can go on to show that e.g. the Diagonalization Lemma and the Gödel-Rosser Theorem which we proved for nice theories apply equally to nice$^+$ ones. And, of course, since the requirement that a nice$^+$ theory includes Q is just there to ensure we are dealing with a recursively adequate theory, our extended theorems can equally well be stated as applying to all consistent, recursively axiomatized, recursively adequate theories. But we won't pause to

spell out any further all the obvious variant ways of reworking our old formal theorems by replacing talk of p.r. axiomatization by recursive axiomatization, and talk of p.r. adequacy by recursive adequacy. It isn't a very illuminating task, and it doesn't lead to any interesting new discoveries.

30.7 Negation-complete theories are recursively decidable

(a) Part (iii) of Theorem 3.1 tells us that if T is an axiomatized formal theory then the set of theorems of T is effectively enumerable. And from that we inferred Theorem 3.2: any consistent, axiomatized, negation-complete formal theory T is decidable. These results involved the informal ideas of an axiomatized theory, of effective enumerability and of decidability. We can now prove counterpart theorems involving the ideas of a recursively axiomatized theory, and of recursive enumerability and recursive decidability.

The counterpart of Theorem 3.1 is evidently

> **Theorem 30.5** *If T is a recursively axiomatized formal theory then the set of theorems of T is r.e.*

So we'll now establish this, and then go on to deduce the formal counterpart of Theorem 3.2, i.e.

> **Theorem 30.6** *Any consistent, recursively axiomatized, negation-complete formal theory T is recursively decidable.*

Proof for Theorem 30.5 We could just take Theorem 3.1 and then use Church's Thesis in labour-saving mode to pass from the informal theorem to the formal one. There is, however, some interest in showing how to do things the harder way (though you are allowed to skip!).

As we said, since T is recursively axiomatized (so the likes of $Axiom_T$ are μ-recursive), there's a μ-recursive relation Prf_T such that $Prf_T(m, n)$ holds when m is the super g.n. of a T-proof of the wff with g.n. n. And we can now give the following definition:[5]

$$\begin{aligned} code(0) &= \mu z[(\exists p < z)(\exists w < z)(z = 2^p \cdot 3^w \wedge Prf_T(p, w)] \\ code(Sn) &= \mu z[(\exists p < z)(\exists w < z)(z > code(n) \wedge \\ &\qquad\qquad z = 2^p \cdot 3^w \wedge Prf_T(p, w)] \end{aligned}$$

The idea here is that, as we run through successive values of n, $code(n)$ runs through all the values of z $(= 2^p \cdot 3^w)$ which code up pairs of numbers p, w such

[5]For brevity, we will apply the minimization operator to a property rather than to its characteristic function – see Section 29.1 (c). We'll also assume without significant loss of generality that T has an unlimited number of theorems: it's left as a boring exercise to cover the exceptional case where T is a hobbled theory with a non-standard logic and only a finite number of consequences.

that $Prf(p, w)$. Assuming T has an infinite number of theorems, the minimization operator in the clause defining $code(Sn)$ always returns a value. So *code* is properly defined by recursion from μ-recursive functions and is μ-recursive.

But now we use the familiar *exp* function from Section 11.8 to extract the values of w from the code numbers z, i.e. to extract the Gödel numbers of theorems. So put

$$enum(n) = exp(code(n), 1)$$

and $enum(n)$, as we want, is μ-recursive, and it enumerates the Gödel numbers of theorems. ⊠

Proof for Theorem 30.6 Again, we could use Church's Thesis in labour-saving mode to pass from our old informal Theorem 3.2 to its formal counterpart. But again we'll do things the harder way (and by all means skip).

Assume T is consistent, recursively axiomatized, and negation complete. We've just established that there is a μ-recursive function *enum* which enumerates the Gödel numbers of T-theorems. Now recall that we can define a p.r. function

$$neg(n) = \ulcorner\neg\urcorner \star n$$

Then if n numbers a wff φ, $neg(n)$ numbers its negation $\neg\varphi$. Hence

if φ is a theorem, then for some j, $enum(j) = \ulcorner\varphi\urcorner$;
if $\neg\varphi$ is a theorem, then for some j, $enum(j) = \ulcorner\neg\varphi\urcorner = neg(\ulcorner\varphi\urcorner)$.

To determine whether the sentence φ is a theorem, we just start evaluating $enum(0)$, $enum(1)$, $enum(2)$, ..., to see which of the values $\ulcorner\varphi\urcorner$, $neg(\ulcorner\varphi\urcorner)$ turns up first on the list of Gödel numbers of theorems. Since T is negation complete and consistent, one and only one of those values will appear on the list. In fact, it will appear at step $\mu j[enum(j) = \ulcorner\varphi\urcorner \vee enum(j) = neg(\ulcorner\varphi\urcorner)]$ in the enumeration. And φ is a theorem just if $\ulcorner\varphi\urcorner$ (rather than $neg(\ulcorner\varphi\urcorner)$) is the value which is listed at that step.

Which means we can put

$thm(m) = 0$ if $enum(\mu j[enum(j) = m \vee enum(j) = neg(m)]) = m$
$thm(m) = 1$ otherwise

And then *thm* is evidently the characteristic function of the property of numbering a T-theorem. But *thm* is easily seen to be μ-recursive (compare Section 11.7 (E)): so we are done. ⊠

(b) Before moving on, let's note an important but simple corollary of Theorem 30.5:

Theorem 30.7 *There are recursively enumerable sets which are not recursively decidable.*

Proof The Gödel numbers of the theorems of any recursively axiomatized theory are r.e.; so the Gödel numbers of PA-theorems in particular are r.e. But

Theorem 30.3 tells us that PA is recursively undecidable. So the set of Gödel numbers of PA theorems is r.e. but not recursively decidable. ⊠

30.8 Recursively adequate theories are not recursively decidable

We are now, as promised, in a position to revisit the main argument of Chapter 6 and construct a formal counterpart for it. So recall: we took Theorem 3.2, the claim that any consistent, axiomatized, negation-complete formal theory T is decidable. We put that together with Theorem 6.1, which says that no consistent, sufficiently strong, axiomatized formal theory of arithmetic is decidable. We then deduced Theorem 6.2: a consistent, sufficiently strong, axiomatized formal theory of arithmetic cannot be negation complete.

We have just proved the formal counterpart of Theorem 3.2, i.e.

> **Theorem 30.6** *Any consistent, recursively axiomatized, negation-complete formal theory T is recursively decidable.*

And we can easily establish the formal counterpart of Theorem 6.1, i.e.

> **Theorem 30.8** *If T is a consistent, recursively axiomatized, recursively adequate theory, then it isn't recursively decidable.*

Proof sketch The argument is exactly parallel to the argument for our slightly less general Theorem 30.3 earlier in this chapter, which said that *nice* theories are recursively undecidable. The proof of *that* theorem invoked Theorem 21.3: no open wff in a nice theory T can capture the corresponding numerical property $Prov_T$. So to prove our new theorem, we simply need to generalize the argument underlying Theorem 21.3 in order to prove: no open wff in a consistent, recursively axiomatized, recursively adequate theory T can capture $Prov_T$. (It is a useful reality check to make sure you understand how to do this, and hence how to complete the proof.) ⊠

Now just put these two theorems together, and we can derive the formal counterpart of our old informal Theorem 6.2:

> **Theorem 30.9** *A consistent, recursively adequate, recursively axiomatized theory of arithmetic cannot be negation complete.*

Which is, of course, just a version of the Gödel-Rosser Theorem minus any information about what the undecidable sentences look like.

30.9 What's next?

Finding a formal counterpart to our informal incompleteness argument of Chapter 6 was straightforward, given what had gone before. So what about giving a formal counterpart of the argument for incompleteness in Chapter 5?

That argument depended on establishing Theorem 5.4: the set of truths of a sufficiently expressive arithmetical language L is not effectively enumerable. Since we now know that Q is recursively adequate, we certainly know that Q's language L_A can express any recursively decidable two-place numerical relation, and hence (by Church's Thesis) is 'sufficiently expressive'. Let True Basic Arithmetic be the set of truths of L_A. Then one formal counterpart to the informal Theorem 5.4 is as follows:

Theorem 30.10 *$\mathcal{T}_A$, True Basic Arithmetic, is not r.e.*

How do we prove this? Well, there are two easy arguments, using materials already to hand:

Proof using Church's Thesis Just take Theorem 5.4 and appeal to Church's Thesis in labour-saving mode. ⊠

Proof sketch using Gödel's Theorem and Craig's Theorem By the corollary of Craig's Theorem, Theorem 19.4, every effectively enumerable set of wffs can be p.r. axiomatized. So an r.e. set of wffs, being effectively enumerable, can be p.r. axiomatized.[6] Hence if $\mathcal{T}_A$ were r.e. it could be p.r. axiomatized by some sound theory T. But then the semantic version of the incompleteness theorem would apply to T, and there would after all be truths in $\mathcal{T}_A$ that are independent of T. Contradiction. ⊠

But, of course, the second proof using the incompleteness theorem is not what we need if we plan to run a counterpart of the Chapter 5 argument, and use Theorem 30.10 itself to *prove* incompleteness! And suppose also that we want a fully worked through proof that doesn't appeal to Church's Thesis as a short cut. Well, looking at Chapter 5 again for a lead, we see that the proof there depended on various intuitive claims about computer programs. Hence, if we are going to sharpen up *that* line of argument and make it rigorous, we'll have to give *some* theoretical treatment of a general-purpose programming framework.

So let's next make a start on doing this. In the last chapters of this book, we'll aim to say *just* enough about Alan Turing's classic analysis of algorithmic computation to throw a little more light on incompleteness phenomena and also to give crucial added support to Church's Thesis which connects the formal idea of μ-recursiveness with the informal ideas of computability/effective enumerability/decidability.

[6] As you'd expect, that formal claim can in fact be proved without going via the informal notion of effective enumerability.

31 Turing machines

In this chapter, we introduce Turing's classic analysis of algorithmic computability.[1] And then – in the next chapter – we will establish the crucial result that the Turing-computable total functions are exactly the μ-recursive functions. This result is fascinating in its own right; it is hugely important historically; and it enables us later to establish some further results about recursiveness and incompleteness in a particularly neat way. So let's dive in without more ado.

31.1 The basic conception

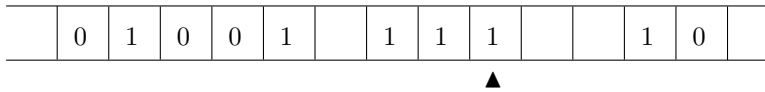
Think of executing an algorithmic computation 'by hand', using pen and paper. We follow strict rules for writing down symbols in various patterns. To keep things tidy, let's write the symbols neatly one-by-one in the squares of some suitable square-ruled paper. Eventually – assuming that we don't find ourselves carrying on generating output for ever – the computation process stops and the result of the computation is left written down in some block of squares on the paper.

Now, Turing suggests, using a two-dimensional grid for writing down the computation is not of the essence. Imagine cutting up the paper into horizontal strips a square deep, and pasting these together into one long tape. We could use that as an equivalent workspace.

Using a rich repertoire of symbols is not of the essence either. Suppose some computational system uses 27 symbols. Number these off using a five-binary-digit code (so the 14th symbol, for example, gets the code '01110'). Then divide each of the original squares on our workspace tape into a row of 5 small cells. Instead of writing one of the original symbols into one of the original big squares, we could just as well write its binary code digit-by-digit into a block of 5 cells.

So – admittedly at some cost in user-friendliness – we can think of our original hand computation as essentially equivalent to following an algorithm (a set of instructions) for doing a computation by writing down or erasing binary digits one-at-a-time in the cells on a linear tape, as in our diagram overleaf. The arrowhead indicates the position of the *scanned cell*, i.e. the one we are examining as we are about to apply the next computational instruction. (We'll assume, by the way, that we can paste on more blank workspace as and when we need it – so, in effect, the tape is unlimited in length in both directions).

[1] See Turing (1936), which is handily reprinted with a very useful long introduction in Copeland (2004).



So much for our workspace. Let's now consider the *instructions* which govern our computation when broken down into minimal steps. We will list these instructions as labelled or numbered lines, where each line tells us what to do depending on the contents of the scanned cell. So we can think of a single line from our algorithm (our set of instructions) as having the form

> q: if the scanned cell contains '0', do action A^0, then go to q^0;
> if the scanned cell contains '1', do action A^1, then go to q^1;
> if the scanned cell is blank, do action A^B, then go to q^B.

where q and the q^j are line-numbers or labels. (That's the general pattern: for a particular line-number q, some of the conditional clauses could be absent.)

It is convenient to distinguish two special labels, 1 and 0: the first will be used to mark the starting instruction of an algorithm, while the second is really a pseudo-label and 'go to 0' will actually mean 'stop the execution of the algorithm'.

What are the possible actions A^j here? There are two basic types. We can *write* in the scanned cell – i.e. over-write any contents in the scanned cell with '0' or '1', or 'write a blank' (i.e. erase the current contents). Or else we can *move* along the tape so that a new cell becomes the scanned cell. We'll take it that any moving is to be done step-wise, one cell at a time. So more complex actions – like e.g. copying the contents of the scanned cell into the next four cells to the right – are to be performed as a sequence of basic actions.

There are therefore five possible minimal actions A, which we can indicate as follows:

> 0: write a '0' in the scanned cell (overwriting any scanned content);
> 1: write a '1';
> B: write a blank;
> L: make the next cell to the left the scanned cell;
> R: make the next cell to the right the scanned cell.

So let's now say, as a first shot, that a *Turing program* is just a collection of instructions of the very simple three-clause form we illustrated, which tell us which action to perform and which line to jump to next.

In executing such a program, we typically start with some number or numbers written as *input* on the work-tape (written in binary digits, of course). And we begin by following the relevant instruction given at the program line labelled 1. We then follow the further instructions we encounter as we are told to jump from line to line. The execution of a set of instructions can then stop for two reasons: we can reach the explicit 'go to 0' instruction to stop; or we can run out of lines to tell us what to do next. In the first case, we'll say that the execution

halts gracefully or simply *halts*; in the second case, we'll say that it *freezes*. If and when the execution halts gracefully, we look at what's left on the tape – and in particular at the block of binary digits containing the scanned cell – and those digits give the numerical *output* of our calculation.

Hence we can say, as a first shot, that a one-place total function f is computed by a Turing program Π if, given n on the tape as input, executing the program eventually yields $f(n)$ as output.

The idea of a Turing computation, then, is extraordinarily simple – it's basically a matter of running a program whose sole programming structure is the 'go to line q' instruction. In Section 31.3, we'll give some mini-examples of Turing programs in operation. But first we really need to refine the ideas we've just sketched. By the end of the next section, then, we will have introduced some sharper terminology and also cleaned up some details. Along the way, there's a number of fairly arbitrary and entirely non-significant choices to be made. We won't comment on these, but don't be surprised to find other choices made in other treatments: the basic conception, however, always essentially the same. (Suggestion: it might help to read the next two sections in parallel, using theory and practice to illuminate each other.)

31.2 Turing computation defined more carefully

(i) *I-quadruples* We define an *instruction-quadruple* (or 'i-quadruple' for short) to be an ordered quadruple of the form

$$\langle q_1, S, A, q_2 \rangle$$

whose elements are as follows:

1. q_1 is a numeral other than '0'; we'll refer to this first element of an i-quadruple as its *label* – to emphasize the point that the numerals here aren't doing arithmetical work. An i-quadruple labelled '1' is an *initial* quadruple.
2. S – representing the contents of the scanned cell – is one of the symbols '0', '1', 'B'. ('B', of course, represents a blank cell.)
3. A is one of the symbols '0', '1', 'B', 'L', 'R': these represent the five possible minimal actions.
4. q_2 is a numeral, pointing to the next instructions to execute.

An i-quadruple is to be read as giving a labelled, conditional, two-part instruction as follows:

q_1: *if* the scanned cell contains S, *do* the action indicated by A, then *go to* the instructions with label q_2 – unless q_2 is '0', in which case halt the execution of the program.

So we can now compress our verbose tri-partite instruction line

> q: if the scanned cell contains '0', do action A^0, then go to q^0;
> if the scanned cell contains '1', do action A^1, then go to q^1;
> if the scanned cell is blank, do action A^B, then go to q^B

into three i-quadruples which share the same initial label, thus:

$$\langle q, 0, A^0, q^0\rangle,\ \langle q, 1, A^1, q^1\rangle,\ \langle q, B, A^B, q^B\rangle.$$

(ii) *Turing programs* Our first shot at characterizing a Turing program therefore comes to this: it is a set of i-quadruples. But we plainly don't want *inconsistent* sets which contain i-quadruples with the same label which issue inconsistent instructions. So let's say more formally:

> A set Π of i-quadruples is consistent if there's no pair of i-quadruples $\langle q_1, S, A, q_2\rangle$, $\langle q_1, S, A', q_2'\rangle$ in Π such that $A \neq A'$ or $q_2 \neq q_2'$.

Which leads to the following sharpened official definition:

> A *Turing program* is a finite consistent set of i-quadruples.[2]

(iii) *Executing Turing programs* We execute a Turing program Π as follows:

1. We start with the work-tape in some *initial configuration* – i.e. with digits occupying a *finite* number of cells, and with some particular cell being scanned. Suppose the content of that initial scanned cell is S.

2. We then look for some appropriate initial quadruple to execute. That is to say, we look for an i-quadruple in Π of the form $\langle 1, S, A, q_2\rangle$: by consistency there is at most one distinct such i-quadruple. We perform action A and jump to the instructions with label q_2.

3. We next look for an i-quadruple in Π of the form $\langle q_2, S', A', q_3\rangle$, where S' is the content of the currently scanned cell. We perform action A' and jump to the instructions with label q_3. And we keep on going ...

4. ... unless and until the execution stops because either (a) we are explicitly told to halt – i.e. we encounter a 'jump to 0' instruction – or (b) the program freezes because there is no relevant i-quadruple for us to apply next.

We will be particularly interested in cases where we run a program starting and finishing in what we'll call *standard* modes. To explain:

[2] Each i-quadruple includes just *one* action instruction A, either a symbol-writing instruction from $\{0, 1, B\}$, or a head-moving instruction from $\{L, R\}$. An obviously equivalent and perhaps neater alternative is to define a program as a set of *i-quintuples*, where each i-quintuple includes *both* a symbol-writing instruction *and* a head-moving instruction.

5. We start with the work-tape in a *standard initial configuration*. In other words, the tape is blank except for containing as input one or more blocks of binary digits, with the blocks separated by single blank cells: and the initial scanned cell is the left-most cell of the left-most block.

6. A run of the program is said to *halt standardly* if (a) it reaches a 'jump to 0' instruction (so the program halts gracefully, rather than just freezes), and (b) it leaves the work-tape blank apart from a single block of binary digits, with (c) the scanned cell being the left-most cell of this block.

Note that if a program halts standardly, it finishes in a state which can serve as a standard initial configuration for other suitable programs. So we can chain together program-modules that start and stop in standard modes to form longer programs.

(iv) *Turing-computable functions* Suppose that $f\colon \mathbb{N} \to \mathbb{N}$, i.e. f is a one-place total numerical function. Then we'll say

> The Turing program Π computes the function f just if, for all n, when Π is executed starting with the tape in a standard initial configuration with n in binary digits on the tape (and nothing else), then the execution halts standardly with the output $f(n)$ in binary digits on the tape.

We can generalize to cover many-place functions. For example, suppose that $g\colon \mathbb{N}^2 \to \mathbb{N}$, i.e. g is a two-place function, which is defined for pairs of numbers. Then,

> The Turing program Π computes the function g just if, for all m, n, when Π is executed starting with the tape in a standard initial configuration with m in binary digits on the tape, followed by a blank cell, followed by n in binary digits (and nothing else), then the execution halts standardly with the output $f(n)$ in binary digits on the tape.

Finally, we say

> A total function is *Turing-computable* if there is a Turing program that computes it.[3]

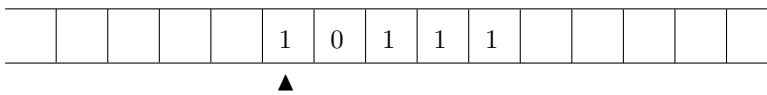
Quite uncontroversially, every Turing-computable function is effectively computable. The claim that the converse is also true, i.e. that every effectively computable total function is Turing-computable is *Turing's Thesis*, which we first met in Section 2.2.

[3]There's a very natural generalization of this definition, where we say that a *partial* function f is Turing-computable if there is a Turing program which computes the right value for input n whenever $f(n)$ is defined and which doesn't halt otherwise. But as we said, we're not going to discuss partial computable functions here.

31.3 Some simple examples

In this section we'll work through three simple example Turing programs, which ought to be enough to illustrate at least the basic ideas. But do feel free to skim through very lightly if you have no taste for this kind of thing.

(a) *The successor function* We'll start with a program that computes the successor function. So suppose the initial configuration of the tape is as follows

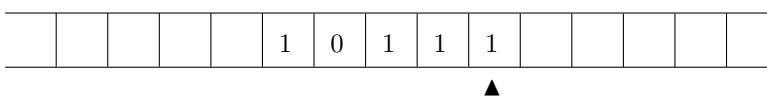


Then the program needs to deliver the result '11000'. The program task can be broken down into three stages:

Stage A We need to make the scanned cell the last cell in the initial block of digits. Executing the following i-quadruples does the trick:

$\langle 1, 0, R, 1\rangle$
$\langle 1, 1, R, 1\rangle$
$\langle 1, B, L, 2\rangle$

These initial instructions move the scanned cell to the right until we overshoot and hit the blank at the end of the initial block of digits; then we shift back one cell, and look for the instructions with label '2':

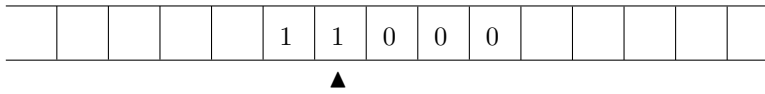


Stage B Now for the core computation of the successor function. Adding one involves putting a '1' in the scanned final cell if it currently contains '0'; or else putting a '0' in the scanned cell if it currently contains '1', and 'carrying one' – i.e. moving the scanned cell one left and adding one again.

The following i-quadruples program for that little routine, and then tell us to look for instructions labelled '4' to finish up:

$\langle 2, 0, 1, 4\rangle$
$\langle 2, 1, 0, 3\rangle$
$\langle 3, 0, L, 2\rangle$
$\langle 2, B, 1, 4\rangle$

Note, the fourth quadruple is to deal with the case where we keep on 'carrying 1' until we hit the blank at the front of the initial block of digits. Executing these instructions gets the tape in our example into the following state:

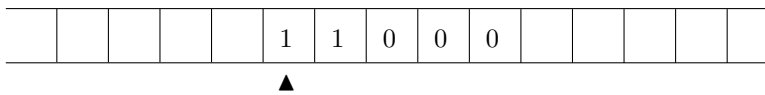


And we now jump to execute the appropriate instruction with label '4'.

Stage C Finishing up. We need to ensure that the scanned cell returns to be at the front of the block, so that the computation halts in standard configuration. Analogously to Stage A, we can write:

$\langle 4, 0, L, 4\rangle$
$\langle 4, 1, L, 4\rangle$
$\langle 4, B, R, 0\rangle$

Following these instructions, the scanned cell moves leftwards until it overshoots the block and then moves it right one cell, and finally the execution halts gracefully:



The scanned cell is now the first in the block of digits. There is nothing else on the tape. So we have halted standardly. Our i-quadruples together give us a program which computes the successor function.

(b) *Another program for the successor function* Note that our successor program, applied to '111', changes those digits to '000', then prefixes a '1' to give the correct output '1000'. So in this case, the output block of digits starts *one cell to the left* of the position of the original input block. We'll now – for future use (in Section 32.2) – describe a variant of the successor program, which this time always neatly yields an output block of digits starting in exactly the *same* cell as the original input block.

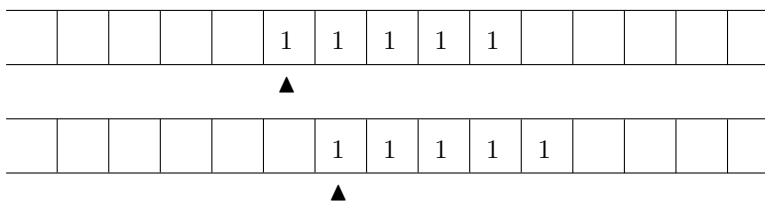
What we need to do, clearly, is to add a routine at the beginning of the program which, if but only if it detects an unbroken block of '1's, shifts that block one cell to the right (by adding a '1' at the end, and deleting a '1' at the beginning). The following will do the trick, and also – like Stage A of our previous program – it moves the current cell to the end of the block:

$\langle 1, 0, R, 12\rangle$
$\langle 1, 1, R, 1\rangle$
$\langle 1, B, 1, 10\rangle$
$\langle 10, 1, L, 10\rangle$
$\langle 10, B, R, 11\rangle$
$\langle 11, 1, B, 11\rangle$
$\langle 11, B, R, 12\rangle$
$\langle 12, 0, R, 12\rangle$

$\langle 12, 1, R, 12\rangle$
$\langle 12, B, L, 2\rangle$

The initial instructions start us off scanning through the input block to the right. If we encounter a '0' then we continue by following the instructions labelled '12' which take the scanned cell to the end of the block. If all we meet are '1's then we put another '1' at the end of the block, and then follow the instructions labelled in the tens, which first take us back to the beginning of the block, then get us to delete the initial '1' and move one cell right, and *then* we follow the instructions labelled '12' to get us to the end of the block

You can check that running through the instructions labelled '1', '10' and '11' changes the state of the tape from the first to the second state illustrated next:



We then just follow the instructions labelled '12', so we end up scanning the cell at the end of the block (and start looking for instructions labelled '2').

We can now add on the same Stage B i-quadruples from our previous successor program to perform the task of adding one, and the same Stage C i-quadruples to get the scanned cell back to the beginning of the resulting block. Putting all those together gives us the desired program.

That's a lot of effort! — but then, programming at the level of 'machine code' (which is in effect what we are doing) *is* hard work. That's why, of course, we ordinarily use high-level programming languages and rely on compilers that work behind the scenes to translate our perspicuous and manageable programs into the necessary instructions for manipulating binary bits.

(c) *A copying program* Our remaining example is a simple program which takes a block of input digits, and produces as output the same block (in the same place), followed by a blank, followed by a duplicate of the original block.

We obviously need to keep track of where we are in the copying process. We can do this by successively deleting a digit in the original block, going to the new block, writing a copy of the deleted digit, returning to the 'hole' we made to mark our place, replacing the deleted digit, and then moving on to copy the next digit. A program for doing all this can be broken down into the following four sub-programs:

1. *Choosing what to do* Examine the scanned cell. If it contains a '0', delete it, and go to sub-program (2). If it contains a '1', delete it, and go to sub-program (3). If it is blank, then we've got to the end of the digits in the

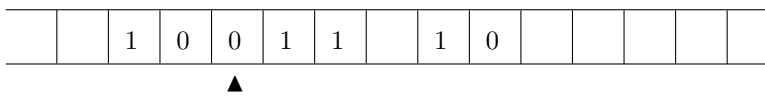
original block, so we just need to call sub-program (4) to ensure that we halt standardly.

2. *Copying a '0'* This routine 'remembers' that we've just deleted a '0'. We scan on to the right until we find the *second* blank – which marks the end of our duplicate block (if and when it exists) – and write a '0'. Then we scan back leftwards until we again find the second blank (the blank created when we deleted the '0'). Rewrite a '0' there, which finishes copying that digit. So now move on to scan the next cell to the right, and return to sub-program (1).

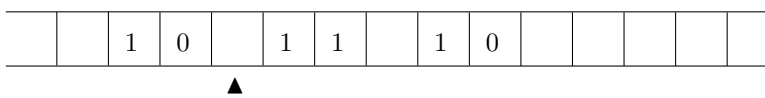
3. *Copying a '1'* Just like sub-program (2), except that this routine 'remembers' that we've just deleted a '1'.

4. *Finish up* Move the scanned cell back to the beginning of the original block.

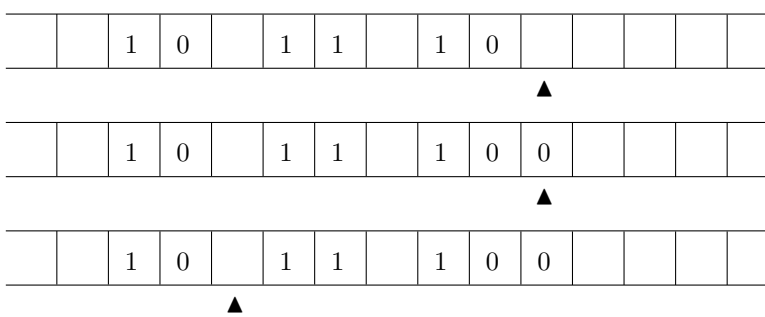
To illustrate, suppose the current state of the tape is like this:

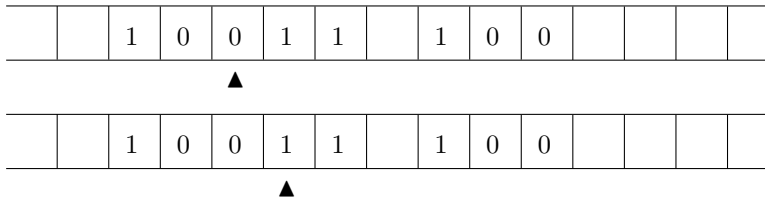


Sub-program (1) instructs us to delete the currently scanned digit and start executing sub-program (2):



Sub-program (2) then takes us through the following stages: (a) we scan to the right to find the end of the second block), (b) we write '0' there, (c) we scan back to the left to find the 'hole' in the first block that we just created, (d) we rewrite '0' there, (e) we move one cell right. So these five stages produce these successive states of the tape:





And we've thereby copied another digit. We keep on going, until we run out of digits in the first block to copy – and then run a 'tidying up' module (4).

Here – just for the fun of it, if that's quite the right word – is how to code up our outlined program strategy into i-quadruples (we've marked the beginnings of the four sub-programs):

$\langle 1, 0, B, 10\rangle$ (1)
$\langle 1, 1, B, 20\rangle$
$\langle 1, B, L, 30\rangle$

$\langle 10, B, R, 11\rangle$ (2)
$\langle 11, 0, R, 11\rangle$
$\langle 11, 1, R, 11\rangle$
$\langle 11, B, R, 12\rangle$
$\langle 12, 0, R, 12\rangle$
$\langle 12, 1, R, 12\rangle$
$\langle 12, B, 0, 13\rangle$
$\langle 13, 0, L, 13\rangle$
$\langle 13, 1, L, 13\rangle$
$\langle 13, B, L, 14\rangle$
$\langle 14, 0, L, 14\rangle$
$\langle 14, 1, L, 14\rangle$
$\langle 14, B, 0, 15\rangle$
$\langle 15, 0, R, 1\rangle$

$\langle 20, B, R, 21\rangle$ (3)
$\langle 21, 0, R, 21\rangle$
$\langle 21, 1, R, 21\rangle$
$\langle 21, B, R, 22\rangle$
$\langle 22, 0, R, 22\rangle$
$\langle 22, 1, R, 22\rangle$
$\langle 22, B, 1, 23\rangle$
$\langle 23, 0, L, 23\rangle$
$\langle 23, 1, L, 23\rangle$
$\langle 23, B, L, 24\rangle$
$\langle 24, 0, L, 24\rangle$
$\langle 24, 1, L, 24\rangle$
$\langle 24, B, 1, 25\rangle$
$\langle 25, 1, R, 1\rangle$

$\langle 30, 0, L, 30\rangle$ (4)
$\langle 30, 1, L, 30\rangle$
$\langle 30, B, R, 0\rangle$

Which all just reinforces the point that writing Turing programs for performing even simple tasks very quickly becomes *very* painful. So we won't give any more detailed examples.[4]

But our concern here isn't really with practical computing but rather with Turing's analysis of what a computation consists in when broken down to its very smallest steps. If he is right that we can treat computations as ultimately symbol manipulation on a 'tape', looking at one cell at a time, etc., then *any genuinely algorithmic step-by-step computation can be replicated using a Turing program.* More on this anon.

[4]Masochists can try their hands at programming one of the many on-line Turing machine simulators which are available (though be careful to read the fine details of how programs are to be specified). However, this is a game that quickly palls!

31.4 'Turing machines' and their 'states'

We have so far imagined a human 'computer' executing a Turing program 'by hand', writing and erasing symbols from a paper tape, mechanically following the program's instructions. Evidently, a machine could do the same job. And any mechanism for running a Turing program might naturally be referred to as a 'Turing machine' (at least if we pretend that its 'tape' is inexhaustible).

But the theory of Turing computations just doesn't care about the hardware implementation. What matters about a Turing machine is always its program. Hence one standard practice is to think of a Turing machine as a type of idealized machine individuated by the Turing program it is running (i.e. same program, same Turing machine: different program, different Turing machine). Another, equally standard, practice is simply to *identify* a Turing machine with its program (it is common enough to read 'A Turing Machine is a set of quadruples such that ...'). Nothing at all hangs on this. When we occasionally talk of Turing machines we'll in fact be thinking of them in the first way. But mostly – for clarity's sake – we'll carry on talking about programs rather than machines.

A final remark. Suppose that a Turing machine (in the first sense) is in the middle of executing a program. It is about to execute some i-quadruple in its program, while scanning a particular cell on a tape which has some configuration of cells filled with digits. We can think of this overall state-of-play as characterized by the 'internal' state of the machine (it is about to execute a quadruple with label q) combined with the 'external' state of the tape (the configuration of the tape, with one cell picked out as the 'current' cell). That's why q-labels are standardly said to identify (internal) *states* of the Turing machine.

32 Turing machines and recursiveness

We are not going to write any more programs to show, case by case, that this or that particular function is Turing-computable, not just because it gets painfully tedious, but because we can now fairly easily establish that *every* μ-recursive function is Turing-computable and, conversely, *every* Turing-computable function is μ-recursive.[1] This equivalence between our two different characterizations of computable functions is of key importance, and we'll be seeing its significance in the remaining chapters.

32.1 μ-Recursiveness entails Turing computability

Every μ-recursive function can be evaluated 'by hand', using pen and paper, prescinding from issues about the *size* of the computation. But we have tried to build into the idea of a Turing computation the essentials of any hand-computation. So we should certainly hope and expect to be able to prove:

Theorem 32.1 *Every μ-recursive function is Turing-computable.*

Proof sketch We'll say that a Turing program is *dextral* (i.e. 'right-handed') if

i. in executing the program – starting by scanning the leftmost of some block(s) of digits – we never have to write in any cell to the *left* of the initial scanned cell (or scan any cell more than one to the left of that initial cell); and

ii. if and when the program halts standardly, the final scanned cell is the *same* cell as the initial scanned cell (in other words, the input block(s) of digits at the beginning of a computation and the final output block start in the same cell).

The key point about a dextral program, then, is that we can run it while storing other data safely on a leftwards portion of the tape, because the program doesn't touch that portion. So complicated computations can proceed by running a series of dextral sub-programs using leftwards portions of the tape to preserve data between sub-programs.

If a function is computable by a dextral Turing program, we'll say it is *dTuring-computable*. Suppose now that the following are all true:

[1] If you happen to be browsing through, not having read the preceding few chapters, or if your attention flickered earlier, I'd better repeat: wherever you see bare talk of a 'function' it means a *total* function. We are not going to be talking about *partial* computable functions.

1. The initial functions are dTuring-computable.
2. If the total functions g and h are dTuring-computable, then so is a function f defined by composition from g and h.
3. If the total functions g and h are dTuring-computable, then so is a function f defined by primitive recursion from g and h.
4. If the recursive function g is dTuring-computable, so is the function f defined from g by regular minimization.

Then take any μ-recursive function f. This can be defined by some chain of definitions by composition and/or primitive recursion and/or regular minimization, beginning with initial functions. So as we follow through f's chain of definitions, we start with initial functions which are dTuring-computable – by (1) – and each successive definitional move takes us from dTuring-computable to dTuring-computable functions – by (2), (3), and (4). So f must be dTuring-computable. So a fortiori, the μ-recursive function f must be plain Turing-computable.

Hence to establish Theorem 32.1, it is enough to establish (1) to (4). But each of those is in fact more or less easy to prove. ⊠

If this overall proof strategy seems familiar, that's because we used the same proof idea e.g. in Chapter 13 when showing that Q is p.r. adequate. The details needed to fill in the proof outline are given in the next section: however, although those details are quite pretty, they are technicalities without much conceptual interest, so by all means skip them.

32.2 μ-Recursiveness entails Turing computability: the details

For enthusiasts, we'll now explain how to fill in the details of our proof-sketch for Theorem 32.1 by establishing points (1) to (4).

Proof sketch for (1) We proved in Section 31.3 (b) that the successor function is not just Turing-computable but is dTuring-computable. It's trivial that the zero function $Z(x)$ is computable by a dextral program – just write a program that takes any block of digits, erases it from the right, and leaves a single '0' on the tape. It's also easily seen that the identity functions are dTuring-computable by erasing and moving blocks of digits. ⊠

Proof sketch for (2) For simplicity, we'll just consider the case of monadic functions. More complex cases can be dealt with using the same basic idea plus a few tricks.

Suppose the program Π_g dTuring computes g, and the program Π_h dTuring computes h. Then to dTuring compute the composite function $f(n) = h(g(n))$, we can just run Π_g on the input n to give $g(n)$, and then run Π_h on that output to calculate $h(g(n))$.

How exactly do we chain together two programs Π_g and Π_h into one composite program? We need to do two things. We first ensure that there is no clash of labels by changing the q-numbers in the i-quadruples in Π_h (doing it systematically, of course, to preserve the all-important cross-references between quadruples). And then we must just ensure that – rather than using the 'halt' label – Π_g ends by telling us to process the first instruction in our re-labelled Π_h. ⊠

Proof sketch for (3) We'll suppose f is defined by the recursion clauses

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

where both g and h are dTuring computable total functions. We need to show that f is dTuring computable. (Simplifying the discussion for the case where the x-variable drops out of the picture, or generalizing it to cover the case where we have an array of variables $\vec{x}$, is straightforward.)

It is convenient to introduce an abbreviated way of representing the contents of a tape. We'll use $\boxed{n}$ to indicate a block of cells containing the binary digits for n, we'll use 'B' to indicate a single blank cell, and then e.g. $\boxed{m}B\boxed{n}$ represents a tape which is blank except for containing m in binary followed by a blank followed by n. So, what we need to describe is a dextral program that takes $\boxed{m}B\boxed{n}$ as input and delivers $\boxed{f(m,n)}$ as output. Here's a sketch:

1. Given the input $\boxed{m}B\boxed{n}$, use a combination of a copying program and a program for subtracting one to get the tape eventually to read as follows:

$$\boxed{m}B\boxed{n}B\boxed{m}B\boxed{n-1}B\boxed{m}B\boxed{n-2}B\ldots$$
$$\ldots B\boxed{m}B\boxed{2}B\boxed{m}B\boxed{1}B\boxed{m}B\boxed{0}B\boxed{m}$$

2. Now move to scan the first cell of the *last* block of digits $\boxed{m}$. Run our program for evaluating g starting in that position. Since this program is by hypothesis dextral, it doesn't visit the portion of the tape any further left than the blank before that last block. *So it is just as if the part of the tape to the left of the last block is completely empty.* Hence the program for evaluating g will run normally on the input m, and it will calculate $g(m)$, i.e. $f(m, 0)$. So after running g the tape will end up reading

$$\boxed{m}B\boxed{n}B\boxed{m}B\boxed{n-1}B\ldots B\boxed{m}B\boxed{1}B\boxed{m}B\boxed{0}B\boxed{f(m,0)}$$

3. Scan the first cell of the concluding three blocks $\boxed{m}B\boxed{0}B\boxed{f(m,0)}$. Run our program for evaluating h starting in that position. Since this program too is by hypothesis dextral, it ignores the leftwards contents of the tape and the program will run normally on the three inputs m, 0, $f(m, 0)$ to calculate $h(m, 0, f(m, 0))$ – i.e. calculate $f(m, 1)$. So when we run h this first time, the tape will end up reading

$$\boxed{m}B\boxed{n}B\boxed{m}B\boxed{n-1}B\ldots B\boxed{m}B\boxed{1}B\boxed{f(m,1)}$$

Now repeat the same operation. So next, scan the first cell of the last three blocks $\boxed{m}\,B\,\boxed{1}\,B\,\boxed{f(m,1)}$). Run the program for evaluating h again, and the tape will end up containing the shorter row of blocks

$$\boxed{m}\,B\,\boxed{n}\,B\,\boxed{m}\,B\,\boxed{n-1}\,B\ldots B\,\boxed{f(m,2)}$$

Keep on going, each time running h using the last three blocks of digits as input, and eventually we will – as desired – be left with just

$$\boxed{f(m,n)}$$

on the tape.

So we've outlined the shape of a program that gives a dTuring computation of the recursively defined f. ⊠

Proof sketch for (4) It remains to show that if the function $g(x,y)$ is dTuring-computable, then so is the function defined by $f(x) = \mu y[g(x,y) = 0]$, assuming g is regular. Hence we want to specify a program that takes $\boxed{m}$ as input and delivers the output $\boxed{\mu y[g(m,y) = 0]}$.

Our task is to run the program g with successive inputs of the form $\boxed{m}\,B\,\boxed{n}$, starting with $n = 0$ and incrementing n by one on each cycle; and we keep on going until $g(m,n)$ gives the output '0', when we return the value of n. But note, we need to do this while 'remembering' at each stage the current values of m and n. So here's a four-stage strategy for doing the job. (Again, simplifying to cover the case where the x-variable drops out of the picture, or generalizing to cover the case where we have an array of variables $\vec{x}$, is straightforward.)

1. We are given $\boxed{m}$ on the tape. Use a modified copier to produce

   $$\boxed{0}\,B\,\boxed{m}\,B\,\boxed{m}\,B\,\boxed{0}$$

   starting at the same point on the tape as the original block. This tape input is now to be fed into Stage (2).

2. Given any input of the kind

   $$\boxed{n}\,B\,\boxed{m}\,B\,\boxed{m}\,B\,\boxed{n}$$

   move to scan the first occupied cell of the second block $\boxed{m}$. Now run the dextral program for g from that starting point, i.e. on the input $\boxed{m}\,B\,\boxed{n}$. By our characterization of Turing programs for functions, g's halts standardly. See whether the output result is 0. If it isn't, go to Stage (3). If it is – and eventually it will be, since g is regular – we finish up with Stage (4).

3. The state of the tape is now

   $$\boxed{n}\,B\,\boxed{m}\,B\,\boxed{g(m,n)}$$

Delete the final block, so we are left with just $\boxed{n}\,B\,\boxed{m}$ on the tape. Next increment the $\boxed{n}$ block by one, and then use a modified copier to yield

$$\boxed{Sn}\,B\,\boxed{m}\,B\,\boxed{m}\,B\,\boxed{Sn}$$

And repeat Stage (2) on this input.

4. The state of the tape is now

$$\boxed{n}\,B\,\boxed{m}\,B\,\boxed{0}$$

We just need to delete the blocks of digits after the first one and we are done!

These stages can clearly be combined into a composite program, which will halt standardly with $\mu y[g(m, y) = 0]$ on the tape. The program is dextral. So the composite program will indeed be a dTuring program for $f(x) = \mu y[g(x, y) = 0]$, and hence, once more, we are done. ⊠.

Which is, all in all, a really rather elegant proof – which is why I just couldn't resist giving it here!

32.3 Turing computability entails μ-recursiveness

As we said, it was only to be hoped and expected that we could show that all μ-recursive functions are Turing-computable. What about the converse claim that the Turing-computable total functions are all μ-recursive?

This is in its way a much more substantial result. For Turing computability involves utterly 'free form' unstructured computation (at the level of 'machine code'): we place no restrictions at all on the way we stack up i-quadruples into a program, other than brute consistency. By contrast, μ-recursive functions are defined in terms of the algorithms that can be described by a higher-level computer language with just 'for' loops and 'do until' loops as the only programming structures. So we might well wonder whether every function which is computable using an arbitrarily organized Turing program can also be computed using only those two types of looping structure. Experience with the versatility of standard computer languages will probably lead us to expect a positive answer: but still, we do need a proof.

So let's now outline the needed argument for

Theorem 32.2 *All Turing-computable functions are μ-recursive.*

Proof strategy Take the case where f is a monadic Turing-computable function. (The argument will generalize in the obvious way to many-place functions.)

Then, by hypothesis, some Turing program Π computes f. In other words, for each n, when Π is run from a standard initial configuration with n on the tape, it halts standardly with $f(n)$ on the tape.

So consider a run of program Π, where we execute the instructions in the next applicable i-quadruple after each tick of the clock: the clock keeps ticking, however, even if the program has halted.

At the j-th tick of the clock, and before the program has halted, the current *state-of-play* of the Turing computation is given by (i) a description of the contents of the tape; (ii) a specification of which cell is the currently scanned cell; (iii) the label for the i-quadruple we will need to execute next. Note that we start with only a finite number of occupied cells on the tape, and each step makes only one modification; so at every step there are still only a finite number of occupied cells; giving the description (i) is therefore always a finite task.

Suppose we use some kind of sensible Gödel-style numbering in order to encode the state-of-play at any step while the program is still running by a single code-number s (where $s > 0$). Then we define *the state-of-play function for* Π:

> $c(n, j) = s$ if the computation which starts with input n is still running at time j; $c(n, j) = 0$ if the computation has halted at time j.

Then we can show that *the state-of-play function for a given* Π *is primitive recursive*. For just reflect that in running through the first j steps of a Turing computation, any searches for the next instruction are bounded by the length of the Turing program Π. So a computation evaluating $c(n, j)$ won't involve open-ended searches, i.e. it can be programmed up using 'for' loops, hence it evaluates a p.r. function. (We won't pause, however, to formally prove this result: the details are just horribly messy and there is neither pleasure nor enlightenment to be had in hacking through them.)

To continue: if the computation halts at step h, then for all $j \leq h$, $c(n, j) > 0$ and $c(n, Sh) = 0$. So $c(n, \mu z[c(n, Sz) = 0])$ gives the code describing the state-of-play – and in particular the contents of the tape – at the point where the computation halts (and by hypothesis, it always does). Therefore,

$$f(n) = \mathit{decode}(c(n, \mu z[c(n, Sz) = 0]))$$

where *decode* is a function that decodes a state-of-play description s and returns the number encoded in the output block of binary digits on the tape at the end of the computation. Assuming the Gödel numbering is sensible so that *decode* is also primitive recursive, it follows immediately that f is μ-recursive. ⊠

If the overall proof strategy here also seems familiar, that's because we used the same basic idea in Section 29.3 when sketching a proof that the Ackermann-Péter function is μ-recursive.

32.4 Generalizing

We've defined a Turing machine as dealing with *binary* symbols (and blanks), using a *one-tape* work-space, moving its focus of operations *one cell* at a time,

and also reading/writing *one cell* at a time. Innumerable variations are evidently possible! For example, we could use a larger repertoire of symbols, or we could consider a machine with more than one tape, or machines than can move the scanned cell by any given number of cells. *But such changes don't make any difference to what our machines can compute* – i.e. they don't take us outside the class of recursive functions. We've already argued for that informally in the case of changing the size of the symbol set (see Section 31.1); and we can similarly argue, case by case, that e.g. working with two tapes doesn't make a difference either, by sketching a way of transforming a program for a two-tape Turing machine into an equivalent Turing program of the original type.

However, such program-transformations can be *very* messy to effect. So note that the proof-strategy of the last section can be adapted to get us much nicer equivalence proofs. For suppose we describe a fancy new machine T. Then, for any sensible machine architecture, the corresponding coding function $c_T(n, j) = s$ will still be primitive recursive (where s is now a suitable code for describing the state-of-play at time j of the computation for input n on our modified machine T). Then by just the same reasoning it follows that the function computed is still μ-recursive, and so what T computes is just what can be computed by some regular Turing machine.

33 Halting problems

Our first main theorem in this chapter establishes the 'recursive unsolvability of the self-halting problem' for Turing machines. This is one of those pivotal results like the Diagonalization Lemma which at first sight seems just an oddity, which is fairly easy to prove, but which entails a whole raft of further key results.

We use this theorem to establish (or re-establish) various claims about incompleteness and decidability. We also prove a version of Kleene's Normal Form Theorem: this leads to yet another proof of incompleteness.

33.1 Two simple results about Turing programs

(a) As a preliminary, let's note

Theorem 33.1 *We can effectively enumerate the Turing programs.*

Proof sketch Use some system for Gödel-numbering sets of i-quadruples. For example, use powers of primes to code up single i-quadruples; then form the super g.n. of a sequence of codes-for-quadruples by using powers of primes again.

Now run through numbers $e = 0, 1, 2, \ldots$. For each e, take prime factors of e, then prime factors of their exponents; and if this reveals that e is the super g.n. of a set of i-quadruples, then check that it is a consistent set and hence a Turing program (that search is bounded by the size of the set of i-quadruples). If e *is* the super g.n. of some Turing program Π, put $\Pi_e = \Pi$; otherwise put $\Pi_e = \varnothing$ (i.e. the empty program with no i-quadruples in it). Then $\Pi_0, \Pi_1, \Pi_2, \ldots$ is an effectively generated list of all possible Turing programs (with repetitions allowed). ⊠

(b) In Section 32.3, we defined $c(n, j)$ as giving a Gödel-style code-number for the state-of-play at the j-th step of a run of a given function-computing program Π with input n; $c(n, j)$ defaults to zero once the run has halted gracefully.

Now we generalize, and introduce the function $c'(e, n, j)$ which gives the code for the state-of-play at the j-th step of a run of the program Π_e in our standard enumeration, where the run starts with input n. But note that previously we were only considering programs that compute total functions, which deliver an output for every input and never freeze. We now need to specify what happens to $c'(e, n, j)$ if the program runs out of instructions. So let's say

> $c'(e, n, j) = s$, where $s > 0$ is the code-number describing the state-of-play as we enter the jth step in a run of the Turing program Π_e given the initial input n, unless either the computation has already

halted gracefully in which case $c'(e, n, j) = 0$, or it has frozen in which case $c'(e, n, j) = c'(e, n, j - 1)$.

Note that, unlike c, the function c' is not regular: for many values of e and n, the program Π_e run on input n will not halt gracefully, so $c'(e, n, j)$ never hits zero as j increases.

We saw, by a rough-and-ready argument, that our original state-of-play function is p.r.: similarly

Theorem 33.2 *The generalized state-of-play function c' is primitive recursive.*

Proof sketch Decoding e to extract the program Π_e doesn't involve any unbounded searches; tracking the execution of Π_e on input n through j ticks of the clock doesn't involve any unbounded searches. So the computation involved in evaluating $c'(e, n, j)$ still involves no unbounded searches overall (the addition of the clause to cope with freezing plainly doesn't change that). So we are still evaluating a p.r. function.[1] ⊠

33.2 The halting problem

(a) Let $\Pi_0, \Pi_1, \Pi_2, \ldots$ be an effective enumeration of Turing machines (identified by their programs). And now let's define

The self-halting problem: to find an effective procedure that will decide, for any e, whether Π_e halts when set to work with its own index-number e as input.

As we'll soon see, there is a whole family of interrelated halting problems. However, before we explain why such problems are interesting, let's immediately prove the following formal theorem:

Theorem 33.3 *The self-halting problem is not recursively solvable.*

Proof Let $h(e)$ be the characteristic function whose value is 0 if the machine number e halts with input e, and is otherwise 1. Then, by definition, the self-halting problem is recursively solvable iff h is μ-recursive. Hence, by last chapter's equivalence theorem, the self-halting problem is recursively solvable iff there is a Turing machine H for computing the function h. But there can be no such machine H.

For suppose otherwise, and consider the result of 'chaining together' H with a simple machine L which, when fed 0 on its input tape, goes into an infinite

[1] The difference between the original sort of state-of-play function c and our generalized function c' can be thought of like this. c is, in effect, a particular Turing program Π 'compiled' into a p.r. function: c' is a p.r. interpreter for arbitrary Turing programs.

loop and never halts but when fed 1 halts leaving the tape untouched. (*Very* easy exercise: write a suitable program for L.) Call this composite machine D.

Like any Turing machine, D will appear in our enumeration of all the machines; it's the machine with program Π_d for some d. So next we ask the cunning question: does D halt when fed its own index number d as input? (And off we go on yet another 'diagonal' argument ...!)

Suppose D *does* halt on input d. Then $h(d) = 0$. Since H, by hypothesis, computes h, this means that H must halt on input d and output 0. So chaining H together with the looper L gives us a composite machine which doesn't halt on input d. But that composite machine is D! So D doesn't halt. Contradiction.

Suppose then that D does *not* halt on input d. That means $h(d) = 1$. Since H computes h, this means that H must halt on input d and output 1. So chaining H together with the looper L gives us a composite machine which halts on input d. That composite machine is D again. So D halts after all. Contradiction.

Hence there can be no such composite machine as D. But L trivially exists. Which means that there can be no such Turing machine as H. ⊠

Now, proving that the self-halting problem is not recursively solvable isn't quite to prove that there is no algorithmic way of telling of an arbitrary Π_e whether it will halt on input e. But, of course, Church's Thesis bridges the gap between the formal and informal claims.

(b) Having proved that the *self*-halting problem isn't solvable, we can go on to deduce that various other halting problems are also unsolvable. The general strategy is to suppose that the target problem P is recursively solvable; show this entails that the self-halting problem is recursively solvable too; then use our last theorem to deduce that P can't be recursively solvable after all; and then we appeal again to Church's Thesis to legitimate interpreting this result as showing that P isn't effectively solvable at all.

To take the simplest but also most important example, consider the problem which is usually simply called

> *The* halting problem: to find an effective procedure that will decide, for any e and n, whether Π_e halts when given input n.

There can't be such an effective procedure, however, because

Theorem 33.3* *The halting problem is not recursively solvable.*

Proof We know from Section 31.3 that there is a Turing copier which takes input e and produces output e, e. Now suppose that there is a Turing machine H' which takes the inputs e, n and delivers a verdict on whether Π_e halts for input n. Chain together the copier and H', and we'd get a composite machine which takes input e and delivers a verdict on whether Π_e halts for input e. But we've just shown that there can be no such machine. So there is no such machine as H'. Hence, the characteristic function $h(e, n)$ – whose value is 0 if the machine

number e halts with input n, and is otherwise 1 – is not μ-recursive. Hence the halting problem isn't recursively solvable. ⊠

(c) Why are halting problems interesting in general? Well, for a start, because when we write Turing (or other) programs we'd like to be able to check that they work as advertised: we'd like to be able to check they don't get stuck in never-ending loops when they are supposed to halt and deliver an output. So it would be good if there were a general effective procedure for program-checking, which can at least determine whether a program Π_e halts when it is supposed to halt. And we've just shown that that's impossible.

33.3 The *Entscheidungsproblem* again

Next, here is a lovely result which shows another aspect of the importance of the unsolvability of the halting problem:

> **Theorem 33.4** *The recursive unsolvability of the halting problem entails that the* Entscheidungsproblem *is recursively unsolvable too.*

Proof Like any p.r. function, c' can be expressed and captured-as-a-function by Q by a four-place open wff $\mathsf{C}(\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{w})$.

So, for any e, $\mathsf{C}(\overline{\mathsf{e}}, \overline{\mathsf{e}}, \overline{\mathsf{j}}, \mathsf{0})$ is true iff $c'(e, e, j) = 0$, i.e. just when Π_e with input e has halted by step j. Hence, since Q is p.r. adequate, $\mathsf{Q} \vdash \mathsf{C}(\overline{\mathsf{e}}, \overline{\mathsf{e}}, \overline{\mathsf{j}}, \mathsf{0})$ whenever Π_e with input e has halted by step j. And for each j, $\mathsf{Q} \vdash \neg\mathsf{C}(\overline{\mathsf{e}}, \overline{\mathsf{e}}, \overline{\mathsf{j}}, \mathsf{0})$ if Π_e never halts with input e.

So now put $\mathsf{H}(\overline{\mathsf{e}}) =_{\text{def}} \exists\mathsf{z}\mathsf{C}(\overline{\mathsf{e}}, \overline{\mathsf{e}}, \mathsf{z}, \mathsf{0})$. It follows that, if Π_e eventually halts at some step j with input e, then $\mathsf{Q} \vdash \mathsf{H}(\overline{\mathsf{e}})$: and if it doesn't ever halt then – assuming Q is ω-consistent (which it is!) – $\mathsf{Q} \nvdash \mathsf{H}(\overline{\mathsf{e}})$.

To put it another way. Let $\hat{\mathsf{Q}}$ again be the conjunction of the seven non-logical axioms of Q. Then $\hat{\mathsf{Q}} \rightarrow \mathsf{H}(\overline{\mathsf{e}})$ will be a logical theorem if and only if Π_e eventually halts with input e.

Now suppose we could recursively decide what's a first-order theorem; then it would follow that we could recursively decide whether Π_e eventually halts with input e, for arbitrary e. Contraposing gives us our theorem.[2] ⊠

33.4 The halting problem and incompleteness

We already knew from Theorem 30.4 that the *Entscheidungsproblem* is not recursively unsolvable. But it is still worth highlighting our new Theorem 33.4,

[2] A link between the halting problem and the *Entscheidungsproblem* was first made by Turing (1936, §11); see also Büchi (1962). However, the original version of the linking argument doesn't go via Q's p.r. adequacy, but depends upon more directly writing claims about programs and halting states as first-order wffs. You can find a textbook presentation of this line of proof in e.g. Boolos et al. (2002, Ch. 11).

just to emphasize how our limitative theorems hereabouts are all so very closely interconnected. This section notes another such interconnection, linking the recursive unsolvability of the halting problem with the book's main topic, Gödelian incompleteness results.

(a) We've just seen that, using a Gödel-numbering scheme to code up facts about Turing machines, there will be a purely arithmetical sentence $\mathsf{H}(\overline{\mathsf{e}})$ which 'says' that the program Π_e halts when run on input e.

Now let's suppose, for reductio, that the truths of L_A are recursively enumerable. Equivalently, suppose that there is a Turing program Π_a – an 'arithmetic machine' – that given successive inputs 0, 1, 2 ... gives as output a listing of (the Gödel numbers of) the truths of L_A. And let's set the following two computational processes going in parallel:

1. Run program Π_e on input e.
2. Set Π_a going, feeding in increasing values of n until it spits out the g.n. for $\neg\mathsf{H}(\overline{\mathsf{e}})$.

If the first process halts, then indeed Π_e halts on input e! And if the second process halts, then the g.n. of $\neg\mathsf{H}(\overline{\mathsf{e}})$ has been spat out by our arithmetic machine, hence $\neg\mathsf{H}(\overline{\mathsf{e}})$ is true, so Π_e *doesn't* halt on input e. Hence, if one or other of our parallel processes must always halt, then running them together in parallel gives us a way of *effectively* solving the self-halting problem – just wait to see which verdict is delivered. And now we can either make a labour-saving appeal to Church's Thesis, or we can fairly easily rigorize this informal line of thought, to get the result that if one of our processes always halts then the self-halting problem would be *recursively* solvable.[3]

But we know that the self-halting problem isn't recursively solvable. So at least for some e, neither of our parallel processes halts. In other words, at least sometimes, (1) Π_e doesn't halt on input e, so $\neg\mathsf{H}(\overline{\mathsf{e}})$ is true; yet it's also the case that (2) Π_a doesn't ever spit out (the g.n. for) $\neg\mathsf{H}(\overline{\mathsf{e}})$, i.e. $\neg\mathsf{H}(\overline{\mathsf{e}})$ isn't in the assumed effective enumeration of all the L_A truths. Contradiction. So there can't be a Turing machine Π_a which enumerates the truths.

Hence, in sum,

> **Theorem 33.5** *The recursive unsolvability of the self-halting problem entails that $\mathcal{T}_A$ – True Basic Arithmetic, the set of truths of L_A – is not r.e.*

(b) So, given that the self-halting problem isn't solvable, that gives us the proof we wanted for Theorem 30.10 – i.e. a proof that $\mathcal{T}_A$ isn't r.e. which depends

[3] Here's one way to rigorize. If one of the processes always halts, then for any e there is a least value of $z = 2^n \cdot 3^j$ such that either (1) $c'(e, e, j) = 0$ or (2) $c'(a, n, j) =$ the code for arriving at the halt state with the g.n. of $\neg\mathsf{H}(\overline{\mathsf{e}})$ on the tape. Then we define a function $h(e)$ to be 0 if $c'(e, e, j) = 0$ for that least value of $z = 2^n \cdot 3^j$ and 1 otherwise, and $h(e)$ will then be a properly defined μ-recursive function which solves the self-halting problem.

on considerations about step-by-step computations.[4] Now combine this with Theorem 30.5 which tells us that the theorems of a recursively axiomatized sound formal theory *are* r.e. Then we can immediately derive

> **Theorem 33.6** *The recursive unsolvability of the self-halting problem entails that a recursively axiomatized sound theory T whose language is L_A can't be negation complete.*

The reasoning for this goes, of course, in a now familiar style: it is exactly parallel to the argument for the informal Theorem 5.7.

33.5 Another incompleteness argument

The alert reader might ask: in proving incompleteness by thinking about Turing machines halting, can we again drop the assumption that we are dealing with a *sound* theory of arithmetic and replace it e.g. with the weaker assumption that we are dealing with an *ω-consistent* theory?

We can! So, merely because it's rather fun if you like this sort of thing, let's re-prove our canonical version of the First Theorem. Here again is what we want to establish:

> **Theorem 17.3** *If T is a p.r. adequate, p.r. axiomatized theory whose language includes L_A, then there is a L_A-sentence φ of Goldbach type such that, if T is consistent then $T \nvdash \varphi$, and if T is ω-consistent then $T \nvdash \neg\varphi$.*

And this time we prove the result without constructing a canonical Gödel sentence or appealing to the Diagonalization Lemma.

Proof Recall the definition of $\mathsf{H}(\overline{\mathsf{e}})$ from Section 33.3, and now consider a Turing machine which, on input e, looks at values of m in turn, and tests whether m numbers a T-proof of $\neg\mathsf{H}(\overline{\mathsf{e}})$, i.e. for successive values of m it evaluates $Prf_T(m, \ulcorner\neg\mathsf{H}(\overline{\mathsf{e}})\urcorner)$, and it eventually halts when this p.r. relation holds or else it trundles on for ever otherwise (evidently, we can program such a machine). This Turing machine, in short, tries to prove that Π_e doesn't halt with input e. For some s, it will be Π_s in our standard enumeration of machines. And – very predictably! – we first ask whether Π_s halts on input s.

If it does, then $T \vdash \mathsf{H}(\overline{\mathsf{s}})$ (because a p.r. adequate theory can prove each true $\mathsf{H}(\overline{\mathsf{e}})$ – see the proof of Theorem 33.4). But also, by the definition of Π_s, its halting implies that for some m, $Prf_T(m, \ulcorner\neg\mathsf{H}(\overline{\mathsf{s}})\urcorner)$, so $T \vdash \neg\mathsf{H}(\overline{\mathsf{s}})$. So, assuming T is consistent, Π_s *doesn't* halt on input s.

And now we can show (i) that if T is consistent, it doesn't prove $\neg\mathsf{H}(\overline{\mathsf{s}})$, and (ii) if T is ω-consistent, then it doesn't prove $\mathsf{H}(\overline{\mathsf{s}})$.

[4] There's also an alternative version that runs more closely parallel to the informal argument in Chapter 5, but we won't spell this out. Enthusiasts should consult Cutland (1980, Ch. 8).

(i) First assume that $T \vdash \neg\mathsf{H}(\bar{\mathsf{s}})$. Then for some m, $Prf_T(m, \ulcorner\neg\mathsf{H}(\bar{\mathsf{s}})\urcorner)$, so Π_s would halt for input s. But we've just shown that Π_s doesn't halt for input s. So T doesn't prove $\neg\mathsf{H}(\bar{\mathsf{s}})$.

(ii) Now assume alternatively that $T \vdash \mathsf{H}(\bar{\mathsf{s}})$ i.e. $T \vdash \exists\mathsf{x}\mathsf{C}(\bar{\mathsf{s}}, \bar{\mathsf{s}}, \mathsf{x}, \mathsf{0})$. But since Π_s doesn't halt on input s, we have $c'(s, s, j) \neq 0$ for each j; and so $T \vdash \neg\mathsf{C}(\bar{\mathsf{s}}, \bar{\mathsf{s}}, \bar{\mathsf{j}}, \mathsf{0})$ for each j. Which makes T ω-inconsistent. So if T is ω-consistent, it doesn't prove $\mathsf{H}(\bar{\mathsf{s}})$.

Finally, you can readily confirm that the undecidable wff $\varphi =_{\text{def}} \neg\mathsf{H}(\bar{\mathsf{s}})$ is of Goldbach type. ⊠

33.6 Kleene's Normal Form Theorem

That's the main business of this rather action-packed chapter done. But I can't resist – now we've got this far – adding a couple more sections. First, in this section, we'll prove a (version) of a very illuminating result due to Kleene (1936a). Then in the final section, we'll show that this result once more gives us another route to incompleteness.

Still – very pretty though the arguments are! – you might want to skip on a first reading.

(a) Review the argument of Section 32.3. By exactly the same reasoning, if the Turing program Π_e in our enumeration in fact *does* compute a μ-recursive function $f_e(n)$ then we will have

$$f_e(n) = decode(c'(e, n, \mu z[c'(e, n, Sz) = 0]))$$

Now just for brevity, let's define two functions by composition as follows:

$$t(x, y, z) =_{\text{def}} c'(x, y, Sz)$$
$$u(x, y, z) =_{\text{def}} decode(c'(x, y, z))$$

Both t and u are p.r. functions: hence – at least, given we've done the hard work of spelling out the definition of c' – we have established the following result:

> There is a pair of three-place p.r. functions t and u such that any one-place μ-recursive function can be given in the standard form
>
> $$f_e(n) = u(e, n, \mu z[t(e, n, z) = 0])$$
>
> for some value of e.

Which is *almost*, but not quite, Kleene's Normal Form theorem.[5]

[5]Note, by the way, that it is emphatically *not* being claimed that for *every* value of e our definition delivers a μ-recursive function. For note that t is not a regular function: for most values of e and n, $t(e, n, z)$ never hits zero. The claim then is only that, for a value of e where f_e *is* μ-recursive, then $t(e, n, z)$ will always hit zero and the equality holds.

Let's note an interesting corollary of almost-Kleene. Imagine for a moment that we could

33.7 Kleene's Theorem entails Gödel's First Theorem

We'll finish this chapter by noting a *wonderfully* pretty result which again neatly takes us back to our most central Gödelian concerns:

> **Theorem 33.8** *Kleene's Normal Form Theorem entails the First Incompleteness Theorem.*

Proof Suppose that there is a p.r. axiomatized formal system of arithmetic S which is p.r. adequate, is ω-consistent (and hence consistent), and is negation complete. Then for every sentence φ either $S \vdash \varphi$ or $S \vdash \neg\varphi$.

Since S is p.r. adequate, there will be a four-place formal predicate T which captures the p.r. function T that appears in Kleene's theorem. And now consider the following definition,

$$\overline{f}_e(n) = \begin{cases} U(\mu z[T(e,n,z)=0]) & \text{if } \exists z(T(e,n,z)=0) \\ 0 & \text{if } S \vdash \forall \mathsf{z} \neg \mathsf{T}(\overline{\mathsf{e}}, \overline{\mathsf{n}}, \mathsf{z}, \mathsf{0}) \end{cases}$$

We'll show that, given our assumptions about S, this well-defines an effectively computable total function for any e.

Take this claim in stages. First, we need to show that our two conditions are exclusive and exhaustive:

i. The two conditions are mutually exclusive (so the double-barrelled definition is consistent). For assume that both (a) $T(e,n,k) = 0$ for some number k, and also (b) $S \vdash \forall \mathsf{z} \neg \mathsf{T}(\overline{\mathsf{e}}, \overline{\mathsf{n}}, \mathsf{z}, \mathsf{0})$. Since the formal predicate T captures T, (a) implies $S \vdash \mathsf{T}(\overline{\mathsf{e}}, \overline{\mathsf{n}}, \overline{\mathsf{k}}, \mathsf{0})$. Which contradicts (b), given that S is consistent.

ii. The two conditions are exhaustive. Suppose the first of them doesn't hold. Then for every k, it isn't the case that $T(e,n,k) = 0$. So for every k, $S \vdash \neg\mathsf{T}(\overline{\mathsf{e}}, \overline{\mathsf{n}}, \overline{\mathsf{k}}, \mathsf{0})$. By hypothesis S is ω-consistent, so we can't also have $S \vdash \exists \mathsf{z} \mathsf{T}(\overline{\mathsf{e}}, \overline{\mathsf{n}}, \mathsf{z}, \mathsf{0})$. Hence by the assumption of negation-completeness we must have $S \vdash \neg\exists \mathsf{z} \mathsf{T}(\overline{\mathsf{e}}, \overline{\mathsf{n}}, \mathsf{z}, \mathsf{0})$, which is equivalent to the second condition.

Which proves that, given our initial assumptions, our conditions well-define a total function $\overline{f}_e$.

---

that every k-place μ-recursive function $f(\vec{n})$ can be given in the form $U(\mu z\,[T_k(e, \vec{n}, z) = 0])$ for some e.

Note, by the way, that we *can* use a similar proof idea to establish Kleene's Normal Form theorem about recursive functions more directly, i.e. without going via the different idea of a Turing program. Just Gödel-number the computations which execute the function's recursive recipe: see e.g. Odifreddi (1999, pp. 90–96).

Finally, I should perhaps just mention – to aid comparisons with other treatments – that Kleene's theorem in its full generality in fact gives a normal form for *partial* computable functions too, if we allow the μ-operator to be applied when it might return no value because there is no least z such that $T(e,n,z) = 0$. But we'll stick to our self-denying ordinance, and not say anything more here about the case of partial functions.

Now we prove that $\overline{f}_e$ is effectively computable. Given values for e and n just start marching through the numbers $k = 0, 1, 2, \ldots$ until we find the first k such that either $T(e, n, k) = 0$ (and then we put $\overline{f}_e(n) = U(\mu z[T(e, n, z) = 0])$), or else k is the super g.n. of a proof in S of $\forall \mathsf{z} \neg \mathsf{T}(\overline{\mathsf{e}}, \overline{\mathsf{n}}, \mathsf{z}, \mathsf{0})$ (and then we put $\overline{f}_e(n) = 0$). Each of those conditions can be effectively checked to see whether it obtains – in the second case because S is p.r. axiomatized, so we can effectively check whether k codes for a sequence of expressions which is indeed an S-proof. And it follows from what we've just shown that eventually one of the conditions must hold.

Two more observations (still with our original assumptions in play):

iii. Suppose f_e is μ-recursive, then $f_e(n) = U(\mu z[T(e, n, z) = 0])$ and the condition $\exists z(T(e, n, z) = 0)$ obtains for every n. And so in that case $f_e = \overline{f}_e$. Hence a list of the $\overline{f}_e$ will include all the μ-recursive functions.

iv. Since, given e, we know how to compute the computable function $\overline{f}_e$, the diagonal function $d(n) =_{\text{def}} \overline{f}_n(n) + 1$ is also effectively computable. But then d is a computable total function distinct from all the $\overline{f}_e$, hence distinct from any μ-recursive function.

So we've just shown that – given our original assumptions – there is a computable total function d which isn't μ-recursive, contradicting Church's Thesis.

Hence, given Church's Thesis, it follows from Kleene's Theorem that, if S is a p.r. axiomatized, p.r. adequate, ω-consistent theory, it can't also be negation complete – which is (the core of) Gödel's First Theorem again, proved this time without any appeal to *Prf* or Prf. ⊠

Since Church's Thesis is here being used in labour-saving mode (to link two formal results together) we can of course give a variant of the argument without it – we just need to turn the informal argument that $\overline{f}_e$ is effectively computable into a formal characterization of $\overline{f}_e$ as a μ-recursive function. But cutting that corner, as our version here does, makes the proof of the theorem more transparent.

And, I'm rather tempted to add, if you don't find it a delight, then maybe you aren't quite cut out for this logic business after all!

34 The Church–Turing Thesis

Right back in Chapter 2 we stated *Turing's Thesis*: a numerical (total) function is effectively computable by some algorithmic routine if and only if it is computable by a Turing machine. Of course, we initially gave almost no explanation of the Thesis. It was only very much later, in Chapter 31, that we developed the idea of a Turing machine and saw the roots of Turing's Thesis in his general analysis of the fundamental constituents of any computation.

Meanwhile, in Chapter 29, we had already introduced the idea of a μ-recursive function and noted the initial plausibility of *Church's Thesis*: a numerical (total) function is effectively computable by an algorithmic routine if and only if it is μ-recursive.

Then finally, in Chapter 32, we outlined the proof that a total function is Turing computable if and only if it is μ-recursive. *Our two Theses are therefore equivalent.*

Given that equivalence, we can now talk of

> *The Church–Turing Thesis* The effectively computable total numerical functions are the μ-recursive/Turing computable functions.

Crucially, this Thesis links what would otherwise be merely technical results about μ-recursiveness/Turing computability with intuitive claims about effective computability; and similarly it links claims about recursive decidability with intuitive claims about effective decidability. For example: it is a technical result that PA is not a recursively decidable theory. But what makes that theorem really significant is that – via the Thesis – we can conclude that there is no intuitively effective procedure for deciding what's a PA theorem. The Thesis is in the background again if, for example, we focus on the generalized version of the First Theorem, which says that any recursively axiomatized ω-consistent theory containing Q is incomplete. Why is *that* significant? Because the Thesis links the idea of being recursively axiomatized to the idea of being a properly axiomatized theory in the informal intuitive sense.

So, in sum, we do depend on the Thesis in giving interesting interpretative glosses to some of our technical results in this book. The Thesis is almost universally accepted. This chapter briefly explains why.

34.1 From Euclid to Hilbert

Any schoolchild, when first learning to calculate (e.g. in learning how to do 'long division'), masters a number of elementary procedures for computing the

answers to various problems. And ever since Euclid, who e.g. gave an elegant and efficient routine for finding the greatest common divisor of two integers, mathematicians have developed more and more algorithms for tackling a wide variety of problems.[1]

These algorithms – to repeat our characterization of Section 2.2 – are finite sequential step-by-step procedures which can be set out in every detail in advance of being applied to any particular case. Every step is (or, by further breaking down, it can be made to be) 'small' so that it is readily executable by a human calculator with limited cognitive resources; the steps don't require any ingenuity or mathematical acumen at all. The rules for moving from one step to the next are entirely determinate and self-contained (e.g. they don't involve pausing to toss coins or to consult an outside oracle).[2] We also require that an algorithmic procedure is to deliver its output after a finite number of computational steps.

Such algorithms are evidently great to have when we can get them. It is no surprise, then, that algorithms to deal with an ever-widening domain of problems have been sought over two millennia. And mathematicians had – we may reasonably suppose – a pretty clear idea of the kind of thing they were looking for before they had ever heard of μ-recursiveness or Turing machines. The idea of an algorithm was simply taken for granted, and wasn't subject to any close analysis. Even in the foundational ferment of the first quarter of the last century, there seems to have been very little explicit discussion. And surely not because the idea was thought too foggy to take very seriously, but for exactly the opposite reason. Compared with the profound worries about the infinite generated by e.g. the set-theoretic paradoxes and the intuitionistic critique of classical mathematics, the idea of a finite, algorithmic, step-by-step procedure must initially have seemed quite clear and unproblematic.

Consider again Hilbert and Ackermann's *Grundzüge der theoretischen Logik* (1928). In §11, as we've noted before, the authors famously pose the *Entscheidungsproblem*, i.e. the problem of deciding for an arbitrary sentence of first-order logic whether it is valid or not. They note that the corresponding decision problem for propositional logic is easily solved – we just use a truth-table: in this

[1] For a generous sample of cases, ancient and more modern, see Chabert (1999).

[2] Determinism is built into the classical conception. Nowadays, we also talk of 'non-deterministic algorithms' – the most exciting cases are the so-called quantum algorithms. This is perhaps a natural enough extension of the original idea; however, it assuredly *is* an extension (though intriguingly, dropping determinism doesn't actually allow us to compute more functions, even if it can radically speed things up). More generally, perhaps it is true to say that

> In fact the notion of algorithm is richer these days than it was in Turing's days. And there are algorithms . . . not covered directly by Turing's analysis, for example, algorithms that interact with their environments, algorithms whose inputs are abstract structures, and geometric or, more generally, non-discrete algorithms. (Blass and Gurevich, 2006, p. 31)

But our concern here is with the articulation of the classical idea of deterministic step-by-step routines of the kind that the founding fathers had in mind in arriving at the concept of an effective procedure.

case, at any rate, we have a 'well-developed algebra of logic' where 'so to speak, an arithmetic treatment' is possible. So now,

> The decision problem for first-order logic presents itself. ... The decision problem is solved when one knows a process which, given a logical expression, permits the determination of its validity/satisfiability in a finite number of operations. ... We want to make it clear that for the solution of the decision problem a process needs to be given by which [validity] can, in principle, be determined (even if the laboriousness of the process would make using it impractical). (Hilbert and Ackermann, 1928, §11)

The quest, then, is for an effective procedure which works – at least in principle, given world enough and time – to deliver a verdict in a finite number of steps by means of a quasi-arithmetical computation. In sum, the *Entscheidungsproblem* – 'the chief problem of mathematical logic' as they call it – is the problem of finding an algorithm for deciding validity. And *Hilbert and Ackermann plainly took themselves to have set a well-posed problem*, which needed only the bare minimum of elucidation which we have just quoted.

Of course, it is one thing to pose a decision problem, and quite another thing to solve it. And there is a crucial asymmetry here between positive and negative cases. Hilbert and Ackermann note, for example, that the decision problem for monadic first-order logic (i.e. logic with only one-place predicates) *does* have a positive solution: see Löwenheim (1915). Now, to show that Löwenheim's procedure solves the restricted decision problem, we just need (i) to see that it is algorithmic, and (ii) to prove that it always delivers the right verdict. And step (i) requires only that we recognize an algorithm when we see one: we don't need e.g. to have command of any general story about necessary conditions for being an algorithm. Similarly for other decision problems with positive solutions.

Suppose, on the other hand, that we begin to suspect that a certain decision problem is unsolvable and want to confirm that conjecture. For example, let's re-run history and suppose that we have tried but failed to discover a general positive solution for the *Entscheidungsproblem*. We find some solutions for limited cases, extending Löwenheim's work, but the suspicion begins to dawn that there isn't a general method for deciding validity for arbitrarily complex sentences. We therefore set out to prove that the decision problem is unsolvable – i.e. prove that there is *no* algorithm which decides first-order validity across the board. Obviously, we can't establish this by an exhaustive search through possible algorithms, one at a time, since there are unlimitedly many of those. Hence, to get our negative proof, we'll need some way of proving facts about the class of all possible algorithms: *now* we will need some general claims about what makes for an algorithm.

In headline terms, then, the situation is this: our implicit pre-theoretical grasp of the idea of a step-by-step computation is enough for a good understanding of what the *Entscheidungsproblem* is. But to show that it is unsolvable we need

more, i.e. we need an explicit story about general features of all algorithmic procedures in order to show that no procedure with those features can solve the problem. Or to quote Kleene:

> [The] intuitive notion of a computation procedure, which is real enough to separate many cases where we know we do have a computation procedure before us from many others where we know we don't have one before us, is vague when we try to extract from it a picture of all possible computable functions. And we must have such a picture, in exact terms, before we can hope to prove that there is no computation procedure at all for a certain function, ... Something more is needed for this. (Kleene, 1967, p. 231)

We'll need to say more in the next chapter, however, about whether 'vague' is *le mot juste* here.

34.2 1936 and all that

Faced with the *Entscheidungsproblem* and the growing suspicion that it was unprovable, Church, Kleene, Gödel and Turing developed a variety of accounts of what makes for an effectively computable function.[3] The drama was played out in Princeton and Cambridge.

(a) To start with, in the early 1930s in Princeton, Alonzo Church and his then students Stephen Kleene and Barkley Rosser developed the so-called 'λ-calculus'. After some false starts, this proved to be a powerful foundational system: Church first conjectured in 1934 that every effectively calculable function is 'λ-definable'.

However, we won't pause to explain what this means. For – at least until it was taken up much later by theoretical computer scientists – the λ-calculus didn't win over a great number of enthusiasts, and a couple of other approaches to computability very quickly came to occupy centre stage.[4]

First, in lectures he gave during his visit to Princeton in 1934, Gödel outlined the idea of a so-called *general recursive* function, drawing on work by Jacques Herbrand. Consider how we compute values of the Ackermann-Péter function p (see Section 29.3). We are given some fundamental equations governing p; and then we work out the value of e.g. $p(2, 1)$ by repeated substitutions in these equations. Roughly speaking, the idea of a general recursive function is the idea of a function whose values can similarly be uniquely generated by repeated

[3] To be absolutely explicit about the connection: the Gödel-numbering trick turns the decision problem about validity into a decision problem about the corresponding numerical property of numbering-a-valid-wff; and the characteristic function trick turns this numerical decision problem into a question about the algorithmic computability of a corresponding function. We'll have to radically truncate the historical story in what follows, but I hope in not too misleading a way. For many more details, see Davis (1982), Gandy (1988) and Sieg (1997).

[4] But see, for example, Trakhtenbrot (1988) and Barendregt (1997).

substitutions into initial equations (where these equations aren't now restricted to primitive recursive definitions but allow e.g. double recursions).

But again we won't go into further details here.[5] For by 1936, Kleene was highlighting the neat concept of a *μ-recursive* function which we met in Chapter 29 – i.e. the idea of a function definable by primitive recursion and minimization. And it almost immediately became clear that in fact the λ-definable total functions are the same functions as Gödel's general recursive total functions which are the same as the μ-recursive functions.[6]

This convergence of approaches was enough to convince Church of the soundness of his original conjecture – a version of Church's Thesis, as we now think of it – identifying calculability with λ-definability. So in his classic 1936 paper, he writes:

> We now define the notion, already discussed, of an effectively calculable function of positive integers by identifying it with the notion of a recursive function of positive integers (or of a λ-definable function of positive integers). This definition is thought to be justified by the considerations which follow [most notably, the equivalence results], so far as a positive justification can ever be obtained for the selection of a formal definition to correspond to an intuitive notion. Church (1936b, p. 356)

Kleene had already been persuaded: he reports that – when Church originally conjectured that the Thesis is true – he 'sat down to disprove it by diagonalizing out of the class of the λ-definable functions'; and finding that this couldn't be done he 'became overnight a supporter'.[7] Gödel, however, was more cautious: his recollection was that he was initially 'not at all convinced' that all effective computations were covered by his characterization of general recursiveness.[8]

(b) Meanwhile, in 1935 in Cambridge, Turing was working quite independently on his account of computation.

We have already sketched this in Section 31.1. To repeat, but now more in his own words, Turing invites us to 'imagine the operations performed by the [human] computer to be split up into "simple operations" which are so elementary that it is not easy to imagine them further divided.'[9] Take our computing agent to be writing on squared paper (though the two-dimensional character of the usual paper we use for hand computation is inessential: we can 'assume that the computation is carried out on one-dimensional paper, i.e. on a tape divided into squares'). Then, Turing continues, we can suppose that a *single* symbol is being observed at any one time: if the computing agent 'wants to observe

[5] See Gödel (1934, §9). For a modern introduction to 'Herbrand-Gödel' computability, as it is also called, see for example Mendelson (1997, §5.5) or Odifreddi (1999, pp. 36–38).

[6] The key equivalence results were published in Kleene (1936a,b).

[7] (Kleene, 1981, p. 59). For the idea of 'diagonalizing out' see Section 29.6.

[8] See the letter from Gödel to Martin Davis as quoted by Kleene (Gödel, 1986, p. 341).

[9] This, and the following quotations in this paragraph, come from Turing (1936, §9).

more, he must use successive observations.'[10] Likewise '[w]e may suppose that in a simple operation not more than one symbol is altered. Any other changes can be split into simple changes of this kind.' Further, we may, without loss of generality, 'assume that the squares whose symbols are changed are always "observed" squares.' Having read and written in some square, we then need to move to work on a new square. The human computer needs to be able to recognize the squares he has to jump to next. And Turing claims that 'it is reasonable to suppose that these can only be squares whose distance from ... the immediately previously observed squares does not exceed a certain fixed amount' (without loss of generality we can suppose the computer gets to new squares stepwise, jumping one square at a time, as in our presentation.) As each simple operation is completed, the human computer moves to some new 'state of mind' which encapsulates determinate instructions about what to do next by way of changing the contents of a square and jumping to a new square (or cell, to revert to our earlier terminology).

In sum,

> It is my contention that these operations [i.e. reading/writing cells on a tape, moving the scanned cell, and jumping to a new instruction] include all those that are used in the computation of a number. (Turing, 1936, §1)

And *if* all that is right, then it is immediate that any function which is effectively computable via an algorithm will be computable by a Turing machine. The converse, however, is surely uncontentious: any Turing computable function is computable by us (given world enough and time), by following the program steps. Whence Turing's Thesis.

(c) Turing learnt of the Princeton research on λ-definability and recursiveness just as he was about to send off his own classic paper for publication. In response, he added a key appendix outlining a proof of the equivalence of Turing-computability with λ-definability. And thus the two strands of work that we've just outlined came together. Still only 24, Turing then left Cambridge to continue his research at Princeton.

(d) It would be wrong to say that in 1936 Church was still defending his Thesis *merely* because a number of different but apparently 'empirically adequate' characterizations of computability had turned out to be equivalent, while in contrast Turing based *his* Thesis on a bottom-up analysis of the very idea of computation. For a start, Church takes over Gödel's 1934 idea of general recursiveness, and that can be thought of as an attempt to locate the essence of a computation in the repeated manipulation of equations. And in his paper, Church also gives

[10] In our presentation, we took the fundamental symbols to be just '0' and '1'. Turing was more generous. But for the reasons we gave, working with (say) 27 basic symbols rather than two in the end makes no odds.

what is often referred to as his 'step-by-step argument' for the Thesis, which is related to the argument which we gave in Section 29.5.

Still, Turing's analysis does seem to dig deeper. It convinced Church: if he was somewhat guarded in 1936 (saying that his own account of computability is justified 'so far as a positive justification can ever be obtained'), he is quite emphatic a year later:

> It is ... immediately clear that computability, so defined [by Turing], can be identified with (especially, is no less general than) the notion of effectiveness as it appears in certain mathematical problems (various forms of the *Entscheidungsproblem*, various problems to find complete sets of invariants in topology, group theory, etc., and in general any problem which concerns the discovery of an algorithm). (Church, 1937, p. 42)

Gödel was similarly convinced. In a 'Postscriptum' added in 1964 to a reprint of his 1934 lectures, he writes:

> Turing's work gives an analysis of the concept of 'mechanical procedure' (alias 'algorithm' or 'computation procedure' or 'finite combinatorial procedure'). This concept is shown to be equivalent with that of a 'Turing machine'. (Gödel, 1934, pp. 369–370).

And Gödel remarks in a footnote that 'previous equivalent definitions of computability' (and he refers to Church's account of λ-definability and his own treatment of general recursiveness) 'are much less suitable for our purpose' – the purpose, that is, of giving a 'precise and unquestionably adequate definition' of concepts like undecidability. The thought, then, is that Turing's analysis *shows* that algorithmic computability is 'unquestionably' Turing computability (while λ-definability and recursiveness earn their keep as alternative characterization of computability because of the equivalence theorems that show them to come to the same as Turing computability).

34.3 What the Church–Turing Thesis is not

To repeat, the Church–Turing Thesis is the claim that the informal notion of an effectively/algorithmically computable numerical function has the same extension as the formal notion of a μ-recursive function.

This Thesis must not be confused (as it too often is) with the entirely different claim that a physical machine can only compute recursive functions – i.e. the claim that any possible computing mechanism (broadly construed) can compute no more than a Turing machine. For perhaps there could be a physical set-up which somehow or other is *not* restricted to delivering a result after a finite number of discrete, deterministic steps, and so is enabled to do more than any Turing machine. Or at least, if such a 'hypercomputer' is impossible, that can't be established merely by arguing for the Church–Turing Thesis.

Let's pause over this important point, and explore it just a little further. We have seen that the *Entscheidungsproblem* can't be solved by a Turing machine. In other words, there is no Turing machine which can be fed (the code for) an arbitrary first-order wff, and which will then decide, in a *finite* number of steps, whether it a valid wff or not. Here, however, is a simple specification for a non-Turing hypercomputer that could be used to decide validity.

Imagine a machine that takes as input the wff ϕ which is to be tested for validity. It then starts effectively enumerating the theorems of the relevant first-order language; this can be done since first-order logic can be treated as an axiomatized formal theory. We'll suppose our computer flashes a light if and when it enumerates a theorem that matches ϕ. Now, our imagined computer *speeds up* as it works. It performs one operation in the first second, a second operation in the next half second, a third in the next quarter second, a fourth in the next eighth of a second, and so on. Hence after two seconds it has done an infinite number of tasks, thereby enumerating and checking *every* theorem to see if it matches ϕ! So if the computer's light flashes within two seconds, ϕ is valid; if not, not. In sum, we can use our wildly accelerating machine to decide validity, because it can go through an *infinite* number of steps in a finite time.

Now, you might very reasonably think that such accelerating machines are a mere philosophers' fantasy, physically impossible and not to be taken seriously. But actually it isn't quite as simple as that. For example, we can describe space-time structures consistent with General Relativity which apparently have the following feature. We could send an 'ordinary' computer on a trajectory towards a spacetime singularity. According to its own time, it's a non-accelerating computer, plodding evenly along, computing for ever and never actually reaching the singularity. But according to us – such are the joys of relativity! – it takes a finite time before it vanishes into the singularity, accelerating as it goes. Suppose we set up our computer to flash us a signal if, as it enumerates the first-order logical theorems, it ever reaches ϕ. We'll then get the signal within a bounded time just in case ϕ is a theorem. So our computer falling towards the singularity can be used to decide validity.

Now, there are quite fascinating complications about whether this fanciful story actually works within General Relativity.[11] But no matter. It is at any rate entirely clear that the issue of whether there could be this sort of Turing-beating physical set-up is *not* settled by the Church–Turing Thesis.

34.4 The status of the Thesis

So the question whether the Church–Turing Thesis is true is *not* an issue about the limits of all possible machines (whatever exactly that means). The question is: are the functions computable-in-principle by step-by-small-step, finite, deterministic processes exactly the recursive/Turing-computable functions? The

[11] For discussion of the ingenious suggestion and its pitfalls, see Earman (1995, Ch. 4).

Church–Turing Thesis gives a positive answer, and no serious challenge has ever been successfully mounted.[12] So the Thesis is almost universally believed. But we have already seen intimations of two rather different assessments of its status, modest and bold.

The more modest view (which perhaps echoes that of the earlier Church) can be expressed as follows:

> Various attempts to characterize the class of effectively computable functions have converged on the same class of recursive functions.[13] No one has ever succeeded in describing a computable function that isn't recursive. All this weight of evidence therefore warrants our adopting recursiveness/Turing-computability at least as an 'explication' of our intuitive concept of effective computability – i.e. as a fruitful, simple, clean *replacement* for our perhaps rather inexact pre-theoretic concept.[14] Hence we can accept the Thesis, though perhaps not so much as a statement of the bald truth as a recommendation about how (uniquely) best to locate a sharply defined class of functions in the area gestured to by our vague informal concept.

The bolder stance (inspired by Turing, and seemingly adopted by the later Church and by Gödel) maintains that

> Analytical reflection on the very notion of an effective calculation shows that the effectively calculable functions can be none other than the recursive/Turing-computable ones, and so the Thesis is a demonstrably true claim about the coextensiveness of our intuitive and formal concepts.

Now, despite Turing's work and Gödel's consequent endorsement of the bolder stance, the modest view seems to have been dominant both in the passing comments of mathematicians and in philosophical discussions of the Church–Turing Thesis. As it happens, I think we *can* be bolder, and explain why in the next chapter. However, don't be distracted by those contentious arguments: *accepting the Thesis in a modest spirit is quite enough for our purposes in this book*. For what we really care about is linking up the technical results about e.g. recursive decidability with claims about what is effectively decidable in the intrinsically interesting intuitive sense. And so long as we accept the Thesis as a working assumption, that's enough to make the link, whatever its philosophical status.

[12] Which isn't to say that there haven't been attempts, and some of these failures can be instructive. See e.g. Kalmár (1959) and the riposte by Kleene (1987).

[13] For accessible reviews of a number of formal definitions of computability in addition to μ-recursiveness and Turing-computability, see Cutland (1980, Chs 1 and 3) and Odifreddi (1999, Ch. 1). Note particularly the idea of a register machine, which idealizes the architecture of a real-world computer on which e.g. C++ programs run.

[14] For more on the idea of an explication, see Carnap (1950, Ch. 1).

35 Proving the Thesis?

An algorithm, we said, is a sequential step-by-step procedure which can be fully specified in advance of being applied to any particular input. Every minimal step is to be 'small' in the sense that it is readily executable by a calculator with limited cognitive resources. The rules for moving from one step to the next must be entirely determinate and self-contained. And an algorithmic procedure is to deliver its output after a finite number of computational steps. The Church–Turing Thesis, as we are interpreting it, is then the claim that a numerical function is effectively computable by such an algorithm iff it is μ-recursive/Turing-computable (note, we continue to focus throughout on *total* functions).

The Thesis, to repeat, is *not* a claim about what computing 'machines' can or can't do. Perhaps there can, at least in principle, be 'machines' that out-compute Turing machines – but if so, such hypercomputing set-ups will not be finitely executing algorithms (see Section 34.3).

And as we also stressed, it is enough for our wider purposes that we accept the Thesis's link between effective computability by an algorithm and μ-recursiveness/Turing computability; we don't have to take a particular stance on the *status* of the Thesis.

But all the same, it is very instructive to see how we might go about following Turing (and perhaps Gödel) in defending a bolder stance by trying to give an informal proof that the intuitive and formal concepts are indeed coextensive. So in this chapter I attempt such a demonstration.

It should be clearly signalled that the core argument is contentious. Still, it is very good to be reminded as we get towards the end of this book that, when it comes to questions about the interpretative gloss that we put on technical results in logic, things aren't always as black-and-white as textbook presentations can make them seem. Take our discussions here as a provocation to further thought and exploration.

35.1 The project

Before proceeding, however, we'd better pause to make it as clear as possible what the project is.

We really need to distinguish *three* levels of concepts which can be in play hereabouts:

1. We start with an initial, inchoate, 'unrefined', concept of *computability* – a concept fixed, insofar as it *is* fixed, by reference to some paradigms of common-or-garden real-world computation.

2. Next there is our idealized though still informal and vaguely framed notion of *effective computability*. Here we require that the computational steps follow an algorithm (in the sense we've just briefly indicated again; see also Section 34.1, fn. 2 and the accompanying text). But we abstract away from 'practical' considerations of how long a computation will take or how much paper will be needed to execute it.

3. Then, thirdly, there are the formal concepts of *μ-recursiveness*, *Turing computability*, and so on.

Now, the move from the first of these notions to the second involves a certain exercise in conceptual sharpening. And there is no doubt an interesting story to be told about the conceptual dynamics involved in reducing the amount of 'open-texture',[1] getting rid of some of the imprecision, in our initial inchoate concept – for this exercise isn't just an *arbitrary* one. However, it plainly would be over-ambitious to claim that in refining our inchoate concept and homing in on the idea of effective computability we are just explaining what we were talking about all along. And that isn't any part of my claim.

Rather, the claim is that, once we have arrived at the second, more refined but still somewhat vague, concept of an *effective* computation, *then* we in fact have a concept which pins down the same unique class of functions as the third-level concepts.

35.2 Vagueness and the idea of computability

(a) Repeatedly in the literature, we find claims like this:

> It is important to realise that the [Church–Turing] thesis is not susceptible of proof: it is an unsubstantiable claim .... (Bridges, 1994, p. 32)

Why? The usual reason given is some version of the following:

> Since our original notion of effective calculability of a function (or of effective decidability of a predicate) is a somewhat vague intuitive one, the thesis cannot be proved. (Kleene, 1952, p. 317)

But what kind of vagueness is in question here, and why is it supposed to block all possibility of a proof?[2]

Let's consider two different ways of elaborating the sort of claim Kleene makes, deploying the ideas of what I'll call *borderline-vagueness-in-extension* and *poly-vagueness* respectively. I will argue that neither interpretation leads to a good argument against the unprovability of the Church-Turing Thesis.

[1] The phrase 'open texture' is due to Friedrich Waismann (1945). Waismann's work has been rather neglected of late; but see Shapiro (2006a) and the Appendix to Shapiro (2006b).

[2] We could readily give twenty quotations making much the same claim.

(b) The concept of a tall man is vague in the sense of allowing the possibility of *borderline cases*, so its extension needn't be sharply bounded. We can line up a series of men, shading from tall to non-tall by imperceptibly tiny stages: there is no sharp boundary to the class of tall men.[3]

Now suppose that it is claimed that the concept of an effectively computable function similarly allows the possibility of borderline cases, so its extension too is not sharply bounded. The picture is that we can likewise line up a series of functions shading gradually from the effectively computable to the non-computable, and again there is no sharp boundary to be found.

If that picture is right, then the extension of 'effectively computable function' is blurry. But the extension of 'recursive function' is of course entirely sharp. So the two concepts don't strictly speaking have the same extension, and the Church–Turing biconditional can't be strictly true.[4]

We have, however, no compelling reason to suppose that this picture *is* right. To be sure, our initial characterization of a step-by-step algorithm used vague talk (e.g. we said that the steps should be 'small'). *But note that it just doesn't follow from the fact that the idea that of an* algorithm *is vague that there can be borderline cases of* algorithmically computable *functions.*

Compare: I wave an arm rather airily and say, 'The men over there are great logicians'. The sense of 'over there' is vague; yet I may determinately refer to none other than Gödel, Church and Kleene, if they are the only men in the vicinity I sweepingly gesture towards (so on any reasonable sharpening of 'over there' in the context, I pick out the same men). Likewise, even if informal talk of effectively computable functions is in some sense an imprecise verbal gesture, it could be the case that this gesture picks out a determinate class of functions (i.e. the only natural class in the vicinity, which is located by any reasonable sharpening of the idea of a function calculable by an algorithm proceeding by small steps). For example, it could be the case that any function presented as computable by a step-by-step process which is a borderline algorithmic process (because the steps are verging on the too 'big') is *also* computable by a Turing machine. And not only could that be the case, but arguably it *is* the case, as we'll see later.

In a slogan, then: vagueness in the *sense* of 'effectively computable' does not necessarily make for vagueness in the *extension* – and the Church–Turing Thesis is a claim about extensions.[5]

[3] I won't here consider the so-called epistemic theory of vagueness according to which there really *is* a sharp boundary to the class of tall men, but we don't and can't know where it is: see the Introduction to Keefe and Smith (1999). Enthusiasts for the epistemic theory will have to rewrite this subsection to accord with their preferred way of thinking of vagueness.

[4] Compare: 'The *extension* of 'effectively computable' is vague and . . . 'recursive' sharpens it, which is the main reason [the Thesis] is important and was introduced by Church in the first place.' (Nelson, 1987, p. 583, my emphasis.)

[5] For us, at any rate, it would be a mistake to write, e.g., 'Church's thesis is the proposal to identify an intuitive notion with a precise, formal, definition' (Folina, 1998, p. 311): it isn't the *notions* but their *extensions* which are being identified.

So there is no simple argument for the supposed borderline vagueness in the extension of 'effectively computable function'. And now we can just be blunt. If that extension did suffer from borderline vagueness, then there could be a series of functions which – as we march along from one function to the next – takes us from the plainly computable-in-principle to the plainly not-computable-even-in-principle via borderline cases. But as far as I know no one has ever purported to describe such a series. I suggest that's because there *isn't* one.

(c) Consider next the discussion of the concept of a polyhedron in Imre Lakatos' wonderful *Proofs and Refutations* (1976). Lakatos imagines a class examining the Euler conjecture that, for any polyhedron, $V - E + F = 2$.[6] And, after some discussion of Cauchy's argument in support of the conjecture, the student Alpha suggests a counterexample. Take a solid cube with a cubic 'hole' buried in the middle of it. This has 12 faces (six outside faces, six inside), 16 vertices, and 24 edges, so in this case $V - E + F = 4$.

Has Alpha described a genuine counterexample? 'Polyhedron' means something along the lines of 'a solid bounded by polygonal faces'. But does Alpha's example count as a solid in the originally intended sense? Well, our pre-theoretical practice arguably just doesn't settle whether Alpha's hollowed-out cube falls under our pre-theoretical concept of a polyhedron or not: so that concept could be sharpened up in different ways, quite consistently with the implicit informal rules we'd mastered for applying and withholding the concept to 'ordinary' cases. To put it another way: Alpha's example reveals that our pre-theoretic talk about polyhedra fails to pick out a single mathematical 'natural kind' – we can legitimately disambiguate the term in more than one way, and we *need* to disambiguate it before we can prove or disprove Euler's conjecture.

Let's say that a term is 'poly-vague' if it ambiguously locates more than one mathematical kind (though each kind might be precisely bounded).[7] And whether or not 'polyhedron' is a good example, the general phenomenon of informal mathematical concepts that can be rigorized in more than one way is surely incontestable. So is 'computable' another one? If it is, then our pre-theoretic talk here fails to pick out a single mathematical 'natural kind', and – before disambiguation – it will therefore be indeterminate what the Church–Turing Thesis says, and hence the Thesis as it stands will be unprovable.

But *is* 'computable function' poly-vague in this way? Well, we have no reason whatsoever to suppose that there is more than one mathematically natural class of total functions in the vicinity picked out by the intuitive notion of a computable function. As we've said before, all our attempts to define computability famously point to exactly the same class of μ-recursive/Turing-computable

[6] V, of course, is the number of vertices, E the number of edges, F the number of faces.

[7] The unfriendly might think it is wrong to think of this as involving any kind of vagueness, properly so called, and that 'confused' would be a better label than 'poly-vague' – see Camp (2002). In a more friendly spirit, you might think that what we are talking about again is a kind of 'open texture' – see Shapiro (2006a). But let's not fuss about how exactly to describe the situation, for the point I need to make isn't sensitively dependent on such details.

functions. So we certainly *can't* take the claim that 'computable function' is poly-vague as a starting point for arguing about the Church–Turing Thesis.

35.3 Formal proofs and informal demonstrations

In sum, we can't take either vagueness-in-extension or poly-vagueness for granted here. So what *are* we to make of Kleene's blunt claim that the idea of an effectively computable function is a vague intuitive one? We seem to be left with nothing much more than the truism that our intuitive, pre-theoretical notion is intuitive and pre-theoretical (i.e. to grasp it doesn't involve grasping an explicit and sharply formulated general theory of what makes for a computation).

So consider again Kleene's expressed pessimism about whether we can 'extract' from our pre-theoretical notion a clearly defined account suitable for theoretical purposes (see the quotation at the end of Section 34.1). Kleene seems to be taking it as obvious that there aren't enough constraints governing our pre-theoretical notion – constraints which anyone who has cottoned on to the notion can be brought to acknowledge – which together with mathematical argument will suffice to establish that the computable functions are the recursive ones. But that claim is *not* obvious. It needs to be defended as the conclusion of some arguments: it can't just be asserted as an unargued presumption.

Here is the same presumption at work again, this time with a couple of added twists:

> How can we in any wise demonstrate [that the intuitively calculable functions are those computable by each of the precise definitions that have been offered]? Ultimately only in some formal system where the vague intuitive concept 'calculable function' would have to be made precise before it could be handled at all, so that in any case the vague intuitive idea would be eliminated before we began, so we would certainly fail to demonstrate anything about it at all. (Steen, 1972, p. 290)

The added assumptions here are that (i) formal systems can't handle vague concepts, and (ii) any genuine demonstration must 'ultimately' be in some formal system.

But neither assumption is compelling. (i) There are in fact a number of competing, non-classical, ways of formally representing the semantics of vague concepts, if we want to be explicit in modelling their vagueness. But waive that point.[8] Much more importantly, (ii) formalization doesn't somehow magically conjure proofs where there were none before. Formalization enforces honesty about what assumptions and inferential rules are being relied on, enabling us to expose suppressed premisses and inferential fallacies, to avoid trading on ambiguities, and so forth. We thereby push to their limits the virtues of explicitness

[8] For some details, see e.g. the editors' Introduction to Keefe and Smith (1999).

and good reasoning that we hope to find in common-or-garden mathematical arguments. But those common-or-garden arguments can perfectly well involve good reasoning and sound demonstrations *before* we go formal.

Here are two examples. First, take the diagonal argument for what we cheerfully labelled Theorem 11.1, i.e. the result that not all effectively computable functions are primitive recursive.[9] Surely, by any reasonable standards, that argument counts as a perfectly good proof, even though one of the concepts it involves is 'a vague intuitive idea'.

Second, take the claim endorsed by almost all those who say that the Church-Turing Thesis is not provable, namely that it would be disprovable if false.[10] The thought is that if we could find a clear case of an intuitively computable function which is provably not recursive, then that would decisively settle the matter. But that again would be a proof involving the application of an intuitive unformalized notion.

The point here is worth highlighting. For note that the argument for Theorem 11.1, for example, didn't invoke mere plausibility considerations. It was, let's say, *a demonstration in informal mathematics*. We might distinguish, then, three levels of mathematical argument – mere plausibility considerations, informal demonstrations, and ideally formalized proofs (or truncated versions thereof).[11] Some philosophers write as if the important divide has ideally formalized proofs on one side, and everything else on the other, pieces of informal mathematics and mere plausibility considerations alike. But that's simply not an attractive view: it misrepresents mathematical practice, and also pretends that the formal/informal distinction is much sharper than it really is.[12] Moreover, it certainly doesn't draw the line in a way that is relevant to our discussion of the status of the Church–Turing Thesis. The question we set ourselves is whether we can do better than give quasi-empirical plausibility considerations. If we *can* support the Thesis

[9] I was hardly stepping out of line in calling this result a 'theorem'! See, for just one example, 'Theorem 3.11' of Cohen (1987, §3.6): 'There is an *intuitively computable* function which is not primitive recursive' (my emphasis).

[10] For dissent, see Folina (1998) – but this again depends on the unargued presumption that any genuine proof must be translatable into a formal proof involving no vague concepts.

[11] One of the nicest examples of a plausibility consideration I know concerns Goldbach's conjecture that every even number greater than 2 is the sum of two primes. Just being told that no one has yet found a counterexample among the even numbers so far examined is not at all persuasive (after all, there are well-known examples of arithmetical claims, e.g. about the proportion of numbers up to n that are primes, which hold up to some utterly enormous number, and then fail). However, if you do a graphical analysis of the distribution of the pairs of primes that add to the successive even numbers you get a remarkable, fractal-like, pattern called the 'Goldbach Comet' which reveals a lot of entirely unexpected structure (I won't reproduce it here: an internet search will reveal some nice examples). And this suddenly makes Goldbach's conjecture look a *lot* more plausible. It no longer looks like a stand-alone oddity, but seems as if it should have interesting interconnections with a rich body of related propositions about the distribution of primes. Though that does indeed make it seem all the more puzzling that the conjecture has resisted proof.

[12] For emphatic resistance to exaggerating the formal/informal distinction, see Mendelson (1990).

using arguments that have something like the demonstrative force of our diagonal argument to show that there are computable functions which aren't p.r., then this is enough to support a *bold* stance on the Thesis (in the sense of Section 34.4).

35.4 Squeezing arguments

(a) We have seen that there is no obvious sense of 'vague' in which it is clear at the very outset both that (i) our informal notion of an effectively computable function is vague and that (ii) this kind of vagueness must prevent our demonstrating that the Church–Turing thesis is true. Which, of course, doesn't show that such a demonstration is possible, but at least it means that we shouldn't give up too soon on the project of looking for one.

So let's consider how we can show that a pre-theoretical, intuitive, concept C *is* co-extensive with some explicitly defined concept E. Here's one key type of argument.

Suppose first that our intuitive understanding of what is *sufficient* for being C enables us to show that, for any suitable α,

1. If α is E, then α is C.

Second, we might also have a conclusive warrant for a claim of the form

2. If α is C, then α is E'.

where E' is another clearly defined formal condition. For our intuitive understanding of what is *necessary* for being C might enable us to show, equivalently, that

2′. If α is *not*-E', then α is *not*-C.

It follows, to put it schematically, that $E \rightarrow C \rightarrow E'$, and we have sandwiched the informal condition of being C between two formally defined conditions.

But now suppose that we can also prove, as a *theorem* relating the two formal concepts, that

3. If α is E', then α is E.

Then this squeezes together the formal conditions that sandwich C, showing that $E \rightarrow C \rightarrow E$. In other words, we can conclude from (1) to (3) that

4. α is C if and only if α is E.

We'll call this type of argument for proving some intuitive concept coextensive with a formal concept a *squeezing argument*.

(b) It might help to have a quick example of this type of argument at work (an example which is important in its own right).[13]

[13]This is a version of an argument famously given by Georg Kreisel (1972). I rather like this illustrative example: but whether you agree will depend, of course, on whether you buy its first two premisses (the third premiss being non-negotiable).

Take the αs to be *arguments couched in a given regimented first-order language*. And let C be the intuitive notion of *being valid-in-virtue-of-form* for such arguments – where an argument α is valid in the classical intuitive sense if, however we reinterpret the relevant non-logical vocabulary, then given α's premisses are true, its conclusion is true too. This intuitive notion might well be counted as 'vague', given that it isn't sharply specified just what counts as a permissible reinterpretation here.[14] However, the constraints on the intuitive notion do in fact suffice to pin down a unique extension for C. Here's how.

Take E to be the property of having a proof in a standard proof system for first-order logic. Then (for any argument α)

1. If α is E, then α is C.

That is to say, the proof system is sound: if you can formally deduce φ from some bunch of premisses Σ, then the inference from Σ to φ is intuitively valid. (Their intuitive soundness is, of course, a principal reason why you accepted the proof system's rules in the first place!)

Second, let's take E' to be the property of *having no countermodel in the natural numbers*. To explain, a countermodel for an argument is an interpretation that makes the premisses true and conclusion false; and a countermodel in the natural numbers is one whose domain of quantification is the natural numbers, where any constants refer to numbers, predicates have sets of numbers as their extensions, and so forth. Then, for any argument α,

2′. If α is *not-*E', then α is *not-*C.

That's because, even if we are a bit foggy about the limits to what counts as an 'interpretation' in the sense used in the intuitive characterization of the idea of validity, we must recognize at least this much: if α does have a countermodel in the natural numbers – i.e. we can reconstrue the argument to be talking about numbers in such a way that the premisses are true and conclusion false – then α certainly can't be valid-in-virtue-of-its-form.

But it's a standard result about first-order logic that

3. If α is E', then α is E.

That is to say, if α has no countermodel in the natural numbers, then α can be deductively warranted.[15]

Putting all that together, we get

[14] Recall: on the standard *formal* notion of an 'interpretation' which we briefly sketched in Section 3.3, we fix on a determinate *set* of objects to be the domain of quantification. Compare the intuitive, pre-theoretic, notion of an interpretation. Are we now allowed to be more relaxed and interpret quantifiers as running over e.g. simply everything, i.e. over a domain which is 'too big' to be a set? Or aren't we?

[15] Recall: the downward Löwenheim-Skolem theorem tells us that if there's any countermodel to α, then there is a countermodel whose domain is some or all the natural numbers. So if α has no countermodel in the natural numbers it can have no countermodel at all, so by completeness is deductively valid.

4. α is C if and only if α is E.

In sum, take the intuitive notion of a first-order inference which is valid in virtue of its form: then our pre-theoretic assumptions about that notion constrain it to be coextensive with a sharply defined formal concept.

Which, I claim, is a very nice result. But whether you buy the details of that argument or not, it illustrates one *sort* of reasoning we might use to argue about computability. So the next question is whether we can turn a similar squeezing trick when the αs are total numerical functions f, C is effective computability, and E is recursiveness/Turing-computability, and hence can demonstrate the Church–Turing Thesis.

35.5 The first premiss for a squeezing argument

The first premiss of a squeezing argument for computability is surely secure.

1. If a function f is E (μ-recursive), then it is C (effectively computable in the intuitive sense).

For if f is μ-recursive it is Turing-computable, so there is a Turing-program which computes it; and a human calculating agent can in principle follow the entirely determinate algorithmic instructions in that program and therefore – given world enough and time – compute the value of f for any given input. So the function is in-principle-computable in the intuitive sense. Which proves the first premiss.

35.6 The other premisses, thanks to Kolmogorov and Uspenskii

Now for the really contentious section! To complete the squeezing argument, we need to find a formally framed condition E' which is weak enough to be a necessary condition for algorithmic computability, yet strong enough to entail μ-recursiveness (remember: we are concentrating on total functions). Can we find such a condition?

(a) With a bit of regimentation, we can think of Turing's epoch-making 1936 paper as gesturing towards a suitable E' and giving us the beginnings of a defence of the second and third premisses for a squeezing argument.

As we've seen, Turing notes various features that would *prevent* a procedure from counting as algorithmic in the intuitive sense. For example, an algorithmic procedure can't require an infinity of distinct fundamental symbols: 'if we were to allow an infinity of symbols, then there would be symbols differing to an arbitrarily small extent' (given they have to be inscribed in finite cells), and then the difference between symbols wouldn't be recognizable by a limited computing

agent. For similar reasons, the computing agent can't 'observe' more than a limited amount of the workspace at one go. And computations shouldn't involve arbitrarily large jumps around the workspace which can't be reduced to a series of smaller jumps – a bound is set by cognitive resources of the computing agent, who needs to be able to recognize where to jump to next.[16]

Now put together the requirements of a finite alphabet, restricted local action on and movement in the workspace, etc. with whatever other similar constraints we can muster. We will get a – formally specifiable – composite necessary condition E' for being calculable by an acceptably algorithmic procedure. In short, we will have something of the form

2. If f is C, then f is E'.

Then, to complete the squeeze, we need to be able to prove that any total function which is E' is computable by a standard Turing machine/is μ-recursive. That is, we need to prove

3. If f is E', then f is E.

Turing's remarks about how we can break down more general kinds of computation into small steps, and hence don't lose generality in concentrating on what we now think of as standard Turing machines (which e.g. change just one cell at a time), can be read as intimating the possibility of this sort of result.

There's now bad news and good news. The bad news is that Turing himself isn't clear about exactly how to fill in the necessary condition E' in order to complete the argument. And in so far as he is clear, his way of spelling out E' is surely too strong to look uncontentious. For example, when we do real-life computations by hand, we often insert temporary pages as and where we need them and equally often throw away temporary working done earlier. In other words, our workspace doesn't have a fixed form: so when we are trying to characterize intuitive constraints on computation we shouldn't assume straight out – as Turing does – that we are dealing with computations where the 'shape' of the workspace stays fixed once and for all.

The good news is that this and other worries can be quieted by appeal to the very general condition for algorithmic computation given in 1958 by A. N. Kolmogorov and V. A. Uspenskii in their paper 'On the definition of an algorithm' (English translation 1963). Note, however, that we do *not* need to accept the claim implicit in their title, namely that the Kolmogorov–Uspenskii (KU) account gives necessary *and* sufficient conditions for being an algorithm: for our purposes, necessity is enough: I'll return to this point.

My claim, then, will be that *when the necessary condition E' is identified with being-computable-by-a-KU-algorithm, the second and third premisses of the Turing-style squeezing argument are demonstrably true.* So in the rest of this section, we will spell out the linked ideas of a KU-algorithm and KU-computability

[16] For more on the same lines, see Turing (1936, §9).

(very slightly modified from Kolmogorov and Uspenskii's original version). We defend premiss (2) with E' understood in terms of KU-computability. We then indicate briefly how the corresponding technical result (3) is proved, and so complete the squeeze.

(b) For Turing's reasons,

i. We still take the alphabet of symbols which any particular algorithm works on to be finite.

But we now start relaxing Turing's assumptions about the shape of the workspace, to get the most general story possible.

First, though, a terminological point. We want a way of referring to the *whole* collection of 'cells' that contain data – whether they are in some general area where computations can take place, or in some reserved areas for passive memory storage ('memory registers'). In the case of Turing machines which lack such reserved areas, it was natural enough to use the term 'workspace' for the whole collection of cells. But now we are moving to a more general setting – and given that 'workspace' naturally contrasts with 'reserved memory' – we better use a different term to make it clear that we intend comprehensive coverage. I suggest *dataspace* – which has the added advantage of having a more abstract ring to it, and so doesn't too readily suggest a simple spatial arrangement.

Since we are going to allow the extent of the dataspace to change as the computation goes along, we can take it to consist of a finite network of 'cells' at every stage, adding more cells as we need them. So,

ii. The dataspace at any stage of the computation consists of a finite collection of 'cells' into which individual symbols are written (we can assume that there is a special 'blank' symbol, so every cell has some content). But now generalizing radically from the 'tape' picture, we'll allow cells to be arranged in any network you like, with the only restriction being that there is some fixed upper limit (which can be different when implementing different algorithms) on the number of immediate 'neighbours' we can get to from any given cell. Being 'neighbours' might be a matter of physical contiguity, but it doesn't have to be: a cell just needs to carry *some* kind of 'pointer' to zero or more other cell(s). Since the algorithm will need to instruct us to operate on cells and certain neighbours and/or move from one cell (or patch of cells) to some particular neighbour(s), we'll need some system for differentially labelling the 'pointers' from a cell to its various neighbours.

For vividness, you can depict cells as vertices in a directed graph, with vertices being linked to their neighbours by 'colour-coded' arrows (i.e. directed edges): the 'colours' are taken from a given finite palette, and the arrows linking one cell to its immediate neighbours are all different colours.[17] And why put an upper

[17] For enthusiasts: here we have slightly modified Kolmogorov and Uspenskii's original treat-

bound on the size of the colour palette (and so put an upper bound on the number of different arrows leading out from a given vertex)? Well, consider a finite computing agent with fixed 'on board' cognitive resources who is trying to follow program instructions of the kind that require recognizing and then operating on or moving around a group of cells linked up by particular arrows: there will be a fixed bound on the number of discriminations the agent can make.

Subject to those constraints, the dataspace can now be structured in any way you like (it needn't be equivalent to a tape or even to an n-dimension 'sheet' of paper): *we only require that the network of cells is locally navigable by a computing agent with fixed cognitive resources.*

Next, we've said all along that an algorithm should proceed by 'small' steps. So we can certainly make the weak requirements that

iii. At every stage in the computation of a particular algorithm, a patch of the dataspace of at most some fixed bounded size is 'active'.

iv. The next step of the computation operates only on the active area, and leaves the rest of the dataspace untouched.

Without loss of generality, we can think of the 'active' area of dataspace at any point in the computation to be the set of cells that are no more than n arrows away from some current focal vertex, where for a given algorithmic procedure n again stays fixed throughout the computation. Why keep n fixed? Because the maximum attention span of a limited cognitive agent stays fixed as he runs the algorithm (he doesn't get smarter!). However, we will otherwise be ultra-liberal and allow the bound n to be as large as you like.[18]

Now for perhaps the crucial radical relaxation of the Turing paradigm:

v. A single computing step allows us to replace a patch of cells in the active area of the dataspace with particular contents and a particular pattern of internal arrows, by a new collection of cells (of bounded size) with new contents and new internal arrows.

So, at least internally to the active area of the dataspace, we can not only fiddle with the contents of the cells at vertices, but change the local arrangement of coloured arrows (while preserving incoming arrows from outside the active area).

---

ment. They treat the dataspace as an *undirected* graph; putting it in our terms, if there is an arrow of colour c from vertex v_1 to v_2, then there is an arrow of colour c from vertex v_2 to v_1. Hence, for them, the restriction to a bounded number of 'arrows out' from a vertex implies a similar restriction to the number of 'arrows in'. But this is unnecessary, and is probably unwelcome if we want maximal generality, as is in effect shown by the independent work in Schönhage (1970, 1980).

[18] At the end of Section 35.7, we note that even this 'fixed bound' constraint can be lifted: but for the moment, we are motivating something like the original KU version. And get the order of the quantifiers right here! – we are only making the weak claim that for any particular algorithm there is some bound on the size of its active dataspace as it runs; we aren't saying that there has to be some one bound which obtains across all algorithms.

As announced, then, the shape of the dataspace itself is changeable – and the dataspace can grow as needed as we replace a small patch by a larger patch with extra cells and new interlinkings.

Having worked on one patch of the dataspace, our algorithm needs to tell us which patch to work on next. And, for Turing's reasons,

vi. There is a fixed bound on how far along the current network of cells the focal vertex of the active patch shifts from one step of the algorithm to the next.

But again, we will be ultra-relaxed about the size of that bound, and allow it to be arbitrarily large for any given algorithm.

Thus far, then, we have described the dataspace and general mode of operation of a KU-algorithm. We now need to define the character of the algorithm itself.

vii. First there can be an initial set of instructions which sets up and structures a patch of dataspace, which 'writes in' the numerical input to the algorithm in some appropriate way. We might, for example, want to set up something with the structure of a register machine, with some initial numerical values in different registers. Or – as with our computation of the Ackermann-Péter function – we might start by writing into the dataspace an equation we are going to manipulate. Rather differently, we might want to start by writing a higher-level program into the dataspace, and then we will use the rest of the algorithm as an interpreter.

viii. The body of a KU-algorithm then consists in a finite consistent set of instructions for changing clumps of cells (both their contents and interlinkings) within the active patch and jumping to the next active patch. Without loss of generality, we can follow Turing in taking these instructions as in effect labelled lines, giving bunches of conditional commands of the form 'if the active patch is of type P, then change it into a patch of type P'/move to make a different patch P'' the new active patch; then go on to execute line q_j/halt'. The instructions are to be implemented sequentially, one line at a time.

ix. Finally, we will need to specify how we read off numerical output from the configuration of cells if and when we receive a 'halt' instruction after a finite number of steps.

We can then offer the following natural definition

x. A (monadic, total) function $f(n)$ is then *KU-computable* if there is some KU-algorithm which, when it operates on n as input, delivers $f(n)$ as output.

The generalization to many-place $f(\vec{n})$ is obvious.

Now, we've given conditions (i) to (x) in a rough and informal style. But it should be clear enough how to go about developing the kind of fully formal abstract characterization articulated in detail by Kolmogorov and Uspenskii. Let's not go into that, however: the important thing is to grasp the basic conception.

(c) Such is the great generality of the KU story, it probably covers *far* too many procedures to count as giving an analysis of the intuitive notion of an algorithm. For what we ordinarily think of as algorithms proceed, we said, by 'small' steps and KU-algorithms can proceed by very large operations on very large chunks of dataspace. But we needn't worry about that at all. The only question we need to focus on is: could the KU story possibly cover *too little*? Well, how could a proposed algorithmic procedure for calculating some function *fail* to be covered by the KU specification?

The KU specification involves a conjunction of requirements (finite alphabet, logically navigable workspace, etc.). So for a proposed algorithmic procedure to fail to be covered, it must falsify one of the conjuncts. But how? By having (and using) an infinite number of primitive symbols? Then it isn't usable by a limited computing agent like us (and we are trying to characterize the idea of an algorithmic procedure of the general type that agents like us could at least in principle deploy). By making use of a different sort of dataspace? But the KU specification only requires that the space has *some* structure which enables the data to be locally navigable by a limited agent. By not keeping the size of active patch of dataspace bounded? But algorithms are supposed to proceed by the repetition of 'small' operations which are readily surveyable by limited agents. By not keeping the jumps from one active patch of dataspace to the next active patch limited? But again, a limited agent couldn't then always jump to the next patch 'in one go' and still know where he was going. By the program that governs the updating of the dataspace having a different form? But KU-algorithms are entirely freeform; there is no more generality to be had.

The claim is that the very modest restrictions on finiteness of alphabet and bounded locality of operation in the dataspace are compulsory for any algorithm: and otherwise, the KU specification imposes no significant restrictions.[19] So, as Kolmogorov and Uspenskii (1963, p. 231) themselves asserted, 'any algorithm is essentially subsumed under the proposed definition'. Hence, as we want:

2. If f is C (effectively computable, by some algorithm), then f is E' (KU-computable).

(d) We need not pause too long over the last premiss of the squeezing argument, i.e. the technical result

3. If f is E' (KU-computable), then f is E (μ-recursive).

[19] That's right, at any rate, given our small modification of their original story, as remarked in the last footnote.

We get this by arithmetization once again – i.e. we use just the same kind of coding argument that we've used in outline twice before, first in Section 29.4 to show that the Ackermann-Péter function is μ-recursive, and then Section 32.3 to show that Turing-computable total functions are all μ-recursive. And indeed, we already implied by a rather handwaving argument in Section 29.5 that this kind of coding argument looked as if it could be used to give a general defence of Church's Thesis. So, in brief:

Sketch of a proof sketch Suppose the KU-algorithm A computes a total monadic function $f(n)$ (generalizing to many-place functions is routine). Define a function $c(n, j)$ whose value suitably codes for the state of play at the j-th step of the run of A with input data n – i.e. the code describes the configuration and contents of the dataspace and locates the active patch. And let the state-of-play code default to zero once the computation has halted, so the computation halts at step number $\mu z[c(n, Sz) = 0]$. Therefore the final state of play of the computation has the code $c(n, \mu z[c(n, Sz) = 0])$. And hence, using a decoding function d to extract the value computed at that state of play, we get

$$f(n) = d(c(n, \mu z[c(n, Sz) = 0]))$$

But d will (fairly trivially) be primitive recursive. So, as usual with this type of proof, all the real work goes into showing that $c(n, j)$ is p.r. too. That result is basically ensured by the fact that the move from one state of play to the next is always boundedly local (so the transition from $c(n, j)$ to $c(n, Sj)$ can be computed using only 'for' loops). Hence $f(n)$ is μ-recursive. ⊠

The squeeze is therefore complete! *We have shown that the (total) computable functions are just the μ-recursive ones.*

(e) A reality check. At the beginning of subsection (c) we said 'Such is the great generality of the KU story, it probably covers far too many procedures to count as giving an analysis of the intuitive notion of an algorithm'. But in subsection (d) we have just argued that the KU-computable functions are exactly the μ-recursive functions and hence, by the squeezing argument, are exactly the effectively computable functions. Those claims might look to be in some tension: it is important to see that they are not.

To repeat, the idea of a *KU-algorithm* may well be too wide to capture the intuitive notion of an algorithm – i.e. more *procedures* count as KU-algorithms than count as mechanical, step-by-small-step, procedures in the ordinary sense. But quite consistently with this, the *KU-computable functions* can be exactly those functions which are algorithmically computable by intuitive standards. That's because – by the third premiss of the squeezing argument – any function that might be computed by a KU-algorithm which operates on 'over-large' chunks of dataspace (i.e. a KU-algorithm which is too wild to count as an intuitive algorithm) is also tamely computable by a standard Turing machine.

35.7 The squeezing argument defended

I will consider three responses, the first two very briefly, the third – due to Robert Black (2000) – at greater length.

(a) One response is to complain that the argument illegitimately pre-empts the possible future development of machines that might exploit relativistic or other yet-to-be-discovered physical mysteries in order to trump Turing machines.

But such complaints are quite beside the point. We need, as before, to sharply distinguish the Church–Turing Thesis proper – which is about what can be computed by finite step-by-step algorithmic procedures – from a claim about what might perhaps be computed by exploiting physical processes structured in some other way.

(b) A second response is to complain that the argument illegitimately pre-empts the future development of mathematics: 'The various familiar definitions of computable functions (e.g. in terms of λ-definability and Turing machines) are radically different one from another. We can't second-guess how future research might go, and predict the new sort of procedures that might be described. So how do we know in advance that another definition won't sometime be offered, which can't be regimented into KU form?'

Well, true, we can't predict how mathematics develops. New paradigms for abstract 'computing' processes may be discovered (mathematicians are always generalizing and abstracting in radical ways). We certainly aren't in the business of second-guessing such developments. We are only making the conceptual point that, whatever ideas emerge they won't count as ideas of algorithmic calculation in the classic sense if they don't cleave to the basic conception of step-by-small-step local manipulations in a dataspace that can be navigated by limited agents. But that conception is all we built into the idea of a KU algorithm. So, to repeat, the claim is: if a procedure is properly describable as algorithmic at all, it will be a KU algorithm.

(c) I am in considerable agreement with Black's excellent discussion; but we part company at the very last step in our responses to Kolmogorov and Uspenskii. Towards the end of his paper, he writes:

> Given the extreme generality of [their] definition of locality, I think it is fair to say that we here have a rigorous proof of Church's thesis *if we can assume that bounded attention span is built into the intuitive notion of effective computation.* However, this is a rather big 'if'. ... [I]t is unclear why the idealization which allows a potentially infinite passive memory (the unlimited supply of Turing-machine tape) should not be accompanied by a corresponding idealization allowing unlimited expansion of the amount of information which can be actively used in a single step. (Black, 2000, p. 256, his emphasis)

However, the reason for not making the second idealization in fact seems pretty clear.

Let's start, though, with a reminder about the first idealization. The sense in which we 'allow a potentially infinite passive memory' is that we are simply *silent* about the extent of the dataspace (other than, in the KU specification, assuming that the occupied space is finite at any stage). In the same way, we are silent about how many steps a successful algorithmic calculation may take. And the reason is the same in both cases. An algorithmic procedure is to be built up from small steps which can be followed by a cognitive agent of limited accomplishment, on the basis of a limited amount of local information (the overall state of the dataspace may well, in the general case, be beyond the agent's ken). Hence, so far as the agent is concerned, the state and extent of the non-active portion of the dataspace at any moment has to be irrelevant, as is the extent of the past (and future!) of the computation. We count an algorithm as in good standing so long as it keeps issuing instructions about what to do locally at the next step (irrespective of the history of the computation or the state of play beyond the active part of the dataspace).

Of course, the real-world implementation of a particular algorithmic procedure may well run up against limitations of external physical resources. But if we can't keep working away at the algorithm because we run out of paper or run out of time, then we – so to speak – blame the poverty of the world's resources rather than say that there is something intrinsically at fault with our step-by-step procedure as an in-principle-computable algorithm.

Now, in the KU story, there is another kind of silence – a silence about just how smart a computing agent is allowed to be. Which means that, for *any* n, we allow KU-algorithms which require the computing agent to have an attention span of 'radius' n (i.e. which require the agent to deal at one go with a patch of cells linked by up to n arrows from some current focal cell). So in that sense, we *do* allow an 'unlimited amount of information' to be 'actively used in a single step'. And to be sure, that looks like a decidedly generous – indeed quite wildly over-generous – interpretation of the idea that an algorithm should proceed by 'small' steps: so we might very well wonder whether every KU-algorithm will be a genuine algorithm in the intuitive sense. But, as we said before, no matter. The claim that concerns us is the converse one that a procedure that *isn't* a KU-algorithm won't count as an algorithm by intuitive standards – a claim which of course gets the more secure the wider we cast the KU net.

What the KU story explicitly *doesn't* allow, though, are procedures which require the capacities of the computing agent to expand in the course of executing a given algorithm (so that on runs of the algorithm for different inputs, the agent has to 'take in at a glance' ever bigger patches of dataspace, without any limit). But isn't that, contrary to Black's talk about 'unlimited expansion', in fact just fine? For it surely goes clean against the whole intuitive idea of an idiot-proof algorithm – which always proceeds by *small* steps, accessible to limited agents – that the steps should get ever bigger, without a particular limit, requiring

ever more cognitive resources from the computer. So Black's worry is arguably quite unfounded: in fact, the *absence* of a limit on the size of the dataspace and the *presence* of a limit on 'attention span' in the KU story are both natural concomitants of the same fundamental assumption that we are dealing with a computing agent of limited internal cognitive capacity, doing limited things locally.

But we can in fact be more concessive to Black. For note that, to establish the argument for the third premiss of the squeezing argument, we just need $c(n, j)$ to be primitive recursive. Suppose, then, that there is a transition function tr that takes us from the code for the j-th step of the computation for input n to the code for the $j + 1$-th step, so $c(n, Sj) = tr(n, j, c(n, j))$. Then $c(n, j)$ will certainly be p.r. if tr is. Now suppose that tr involves the use of a restricted search operator $(\mu z \leq a(n, j))$, where $a(n, j)$ – whose value is to be extracted from $c(n, j)$ – is a p.r. function giving the size of the active patch of dataspace that needs to be looked around in applying a transition rule for the next step. *Then tr could still be p.r. even if $a(n, j)$ grows primitively recursively.* So in fact we could tweak the definition of a KU-algorithm to allow the size of the active patch to grow, as fast as you like so long the growth is governed by some primitively recursive function, and yet c would *still* remain p.r., as needed for the squeezing argument.

Which should be more than enough to quiet Black's worries.

35.8 To summarize

So I've argued that the Church–Turing Thesis, which links some of our earlier technical results to intuitive claims about axiomatized theories, decidability, etc., itself seems susceptible to an intuitively compelling demonstration.

But be that as it may. Let's stress the key point again. Whatever its exact status, the Thesis is quite secure enough for us to lean on: and *that*'s sufficient for all our technical results earlier in the book to have their advertised deep interest.

36 Looking back

Let's finish by taking stock one last time. At the end of the last Interlude, we gave a road-map for the final part of the book. So we won't repeat the gist of that detailed local guide to recent chapters; instead, we'll stand further back and give a global overview. And let's concentrate on the relationship between our various proofs of incompleteness. Think of the book, then, as falling into three main parts:

(a) The first part (Chapters 1 to 7), after explaining various key concepts, proves two surprisingly easy incompleteness theorems. Theorem 5.7 tells us that if T is a *sound* axiomatized theory whose language is *sufficiently expressive*, then T can't be negation complete. And Theorem 6.2 tells us that we can weaken the soundness condition and require only consistency if we strengthen the other condition (from one about what T can express to one about what it can prove): if T is a *consistent* axiomatized theory which is *sufficiently strong*, then T again can't be negation complete.

Here the ideas of being sufficiently expressive/sufficiently strong are defined in terms of expressing/capturing enough effectively decidable numerical properties or relations. So the *arguments* for our two initial incompleteness theorems depend on a number of natural assumptions about the intuitive idea of effective decidability. And the *interest* of those theorems depends on the assumption that being sufficiently expressive/sufficiently strong is a plausible desideratum on formalized arithmetics. If you buy those assumptions – and they are intuitively attractive ones – then we have proved Gödelian incompleteness without tears (and incidentally, without having to construct any 'self-referential' sentences). But it isn't very satisfactory to leave things like that, given the undischarged assumptions. And much of the ensuing hard work over the nearly three hundred pages that follow is concerned with avoiding those assumptions, one way or another.

(b) The second part of the book (Chapters 8 to 28) proves incompleteness again, without relying on informal assumptions about a theory's being sufficiently expressive/sufficiently strong, and without relying on the idea of effective decidability at all. Two ideas now drive the proofs – ideas that are simple to state but inevitably rather messy to prove. (i) The first idea involves the *arithmetization of syntax* by Gödel-numbering: we show that key numerical relations like *m codes for a* PA *proof of n* are primitive recursive, and similarly for any p.r. axiomatized theory. (ii) The other idea is that PA and even Q are *p.r. adequate* theories: that is to say, they can express/capture all p.r. functions

and relations. With these two ideas in place, the rest of the argument is then relatively straightforward. We can use Gödel's method of explicitly constructing a particular sentence G which, via coding, 'says' of itself that it isn't provable. Or, what comes to much the same, we can prove the Diagonalization Lemma and hence that there *is* a Gödel sentence, in the sense of a fixed point for $\neg\mathsf{Prov}(x)$, even if we don't explicitly construct one. Either way, we can then show that PA is incomplete if sound, since it can't prove its Gödel sentences. And again, we can weaken the semantic assumption of soundness to a syntactic assumption, this time the assumption of ω-consistency. If PA is ω-consistent, then it is incomplete.

Further, the dual arguments for PA's incompleteness then generalize in easy ways. By the semantic argument, *any* sound theory which contains Q and which is sensibly axiomatized (more carefully: is p.r. axiomatized) is incomplete: moreover there are undecidable sentences which are Π_1 sentences of arithmetic. In fact, by the syntactic argument, *any* ω-consistent p.r. axiomatized theory which contains Q is incomplete, whether it is sound or not. Or, to generalize to the result closest to Gödel's own First Theorem, Theorem 17.3: if T includes the language of basic arithmetic, can capture all p.r. functions, is p.r. axiomatized, and is ω-consistent, then there are Π_1 arithmetical sentences undecidable in T. And to improve these last results, Rosser's construction then tells us how to replace the assumption of ω-consistency with plain consistency.

The Second Theorem then reveals that there are undecidable sentences like Con_T which aren't 'self-referential' either, thus reinforcing a point that emerged in the first part of the book: incompleteness results aren't somehow irredeemably tainted with self-referential paradox. And where the First Theorem sabotages logicist ambitions, the Second Theorem sabotages Hilbert's Programme.

(c) The final part of the book (Chapters 29 to 35) returns to the approach to incompleteness we initially explored in the first part. But we now trade in the informal notion of effective decidability for the idea of *recursive* decidability – or what provably comes to the same, for the idea of being-decidable-by-a-Turing-machine (the Church–Turing Thesis tells us that this is a good trade). We can then use results about Turing machines to re-prove incompleteness, still using Cantor-like diagonalization tricks but now without going via the Diagonalization Lemma, to get formal analogues of our informal theorems Theorem 5.7 and Theorem 6.2. And finally, for fun, we also proved the First Theorem again by a new route, by invoking Kleene's Normal Form Theorem and the Church–Turing Thesis that whatever is effectively computable is Turing computable.

So that all gives us a number of different routes to the pivotal Gödelian incompleteness results. Our discussions are certainly not the end of the story: there are other routes too, some of a different character again, and not involving any kind of diagonalization tricks. They must remain a story for another day. But at least we've made a start ...

Further reading

(a) Let's begin with four recommendations for parallel reading (fuller publication details are in the bibliography).

1. For a beautifully clear introduction, presenting and rounding out some of the logical background we assume in this book, and also giving a very nice proof of incompleteness, Christopher Leary's *A Friendly Introduction to Mathematical Logic* is hard to beat.

2. George Boolos and Richard Jeffrey's *Computability and Logic* (3rd edition) covers most of the same ground as this book, and more besides, but does things in a different order – dealing with the general theory of computability before exploring Gödelian matters. There is also a significantly expanded fourth edition, with John Burgess as a third author: but many readers may prefer the shorter earlier version.

3. Raymond Smullyan's terse and very elegant *Gödel's Incompleteness Theorems* is deservedly a modern classic.

4. Even briefer is the *Handbook* essay on 'The Incompleteness Theorems' by Craig Smoryński, though it still manages to touch on some issues beyond the scope of this book.

(b) So where next? There are many pointers in the footnotes scattered through this book. So I'll confine myself here to mentioning readily available books which strike me as being, in their different ways, particularly good. First, a group very directly concerned with Gödelian issues:

5. Raymond Smullyan's *Diagonalization and Self-Reference* examines in detail exactly what the title suggests.

6. In his *The Logic of Provability*, George Boolos explores in depth the logic of the provability predicate (the modal logic of our '$\Box$').

7. For more on what happens when we add sequences of consistency statements to expand an incomplete theory, and much else besides, see Torkel Franzén, *Inexhaustibility*.

Next, a couple of fine books that explore Peano Arithmetic and its variants in more depth:

8. Petr Hájek and Pavel Pudlák's *Metamathematics of First-Order Arithmetic* is encyclopaedic but still accessible.

9. Richard Kaye's *Models of Peano Arithmetic* will tell you – among many other things – about 'natural' mathematical statements which are independent of PA.

As we saw in the last part of the book, incompleteness results are intimately related to more general issues about computability and decidability. For more on those issues, here are a few suggestions:

10. For a brief and very accessible overview, see A. Shen and N. K. Vereshchagin, *Computable Functions*.

11. Nigel Cutland's *Computability* is deservedly a modern classic, with much more detail yet also remaining particularly accessible.

12. A more recent text, also quite excellent, is S. Barry Cooper, *Computability Theory*.

13. Another splendid book – with more historical and conceptual asides than the others – is Piergiorgio Odifreddi, *Classical Recursion Theory*.

14. For much more on the Church-Turing Thesis, though of variable quality, see Olszewski et al. (eds), *Church's Thesis after 70 Years*.

Finally, we mentioned second-order arithmetics in Chapter 22. For more on second-order theories, see the indispensable

15. Stewart Shapiro, *Foundations without Foundationalism*.

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 page 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.

Avigad, J., 2003. Number theory and elementary arithmetic. *Philosophia Mathematica*, 11: 257–284.

Balaguer, M., 1998. *Platonism and Anti-Platonism in Mathematics*. New York: Oxford University Press.

Barendregt, H., 1997. The impact of the Lambda Calculus in logic and computer science. *Bulletin of Symbolic Logic*, 3: 181–215.

Benacerraf, P., 1967. God, the Devil and Gödel. *The Monist*, 51: 9–32.

Bernays, P., 1930. The philosophy of mathematics and Hilbertian proof-theory. In Mancosu 1998, pp. 234–265.

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.

Blass, A. and Gurevich, Y., 2006. Algorithms: a quest for absolute definitions. In Olszewski et al. (2006), pp. 24–57.

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

Boolos, G., 1998. *Logic, Logic, and Logic*. Cambridge, MA: Harvard University Press.

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

Boolos, G. and Jeffrey, R., 1989. *Computability and Logic*. Cambridge: Cambridge University Press, 3rd edn.

Bridge, J., 1977. *Beginning Model Theory*. Oxford Logic Guides 1. Oxford: Clarendon Press.

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., 1994. On Gödel's theorems on lengths of proofs I: number of lines and speedups for arithmetic. *Journal of Symbolic Logic*, 39: 737–756.

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

Camp, J. L., 2002. *Confusion: A Study in the Theory of Knowledge*. Cambridge, MA: Harvard University Press.

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.

Chihara, C. S., 1973. *Ontology and the Vicious Circle Principle*. Ithaca: Cornell University Press.

Church, A., 1936a. A note on the *Entscheidungsproblem*. *Journal of Symbolic Logic*, 1: 40–41.

Church, A., 1936b. 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*, 7: 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 Zahlen? In Ewald 1996, Vol. 2, pp. 790–833.

Detlefsen, M., 1986. *Hilbert's Program*. Dordrecht: D. Reidel.

Earman, J., 1995. *Bangs, Crunches, Whimpers, and Shrieks: Singularities and Acausalities in Relativistic Spacetimes*. New York: Oxford University Press.

Enderton, H. B., 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.

Fairtlough, M. and Wainer, S. S., 1998. Hierarchies of provably recursive functions. In S. R. Buss (ed.), *Handbook of Proof Theory*, pp. 149–207. Amsterdam: Elsevier Science B.V.

Feferman, S., 1960. Arithmetization of metamathematics in a general setting. *Fundamenta Mathematicae*, 49: 35–92.

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 Logic*. New York: Oxford University Press.

Feferman, S., 2000. Why the programs for new axioms need to be questioned. *Bulletin of Symbolic Logic*, 6: 401–413.

Feferman, S., 2006. Are there absolutely unsolvable problems? Gödel's dichotomy. *Philosophia Mathematica*, 14: 134–152.

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., 2004. *Inexhaustibility: a Non-Exhaustive Treatment*. Association for Symbolic Logic: Lecture Notes in Logic 16. Wellesley, MA: A. K. Peters.

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.

Frege, G., 1984. *Collected Papers*. Oxford: Basil Blackwell.

Friedman, H., 2000. Normal mathematics will need new axioms. *Bulletin of Symbolic Logic*, 6: 434–336.

Gallier, J. H., 1991. What's so special about Kruskal's theorem and the ordinal Γ_0? A survey of some results in proof theory. *Annals of Pure and Applied Logic*, 53: 199–260.

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 Szabo 1969.

Gentzen, G., 1936. Die Widerspruchsfreiheit der reinen Zahlentheorie. *Mathematische Annalen*, 112: 493–565. Translated as 'The consistency of elementary number theory' in Szabo 1969.

Gentzen, G., 1937. Der Unendlichkeitsbegriff in der Mathematik. Unpublished lecture, translated as 'The concept of infinite in mathematics' in Szabo 1969.

Gentzen, G., 1938. Neue Fassung des Widerspruchsfreiheitsbeweises für die reine Zahlentheorie. *Forschungen zur Logik*, 4: 19–44. Translated as 'New version of the consistency proof for elementary number theory' in Szabo 1969.

Giaquinto, M., 2002. *The Search for Certainty*. Oxford: Clarendon Press.

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., 1933. The present situation in the foundations of mathematics. In Gödel 1995, pp. 45–53.

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., 1951. Some basic theorems on the foundations of mathematics and their implications. In Gödel 1995, pp. 304–323.

Gödel, K., 1958. On a hitherto unutilized extension of the finitary standpoint. In Gödel 1990, pp. 241–251.

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., 2003a. *Collected Works, Vol. 4: Correspondence A–G*. Oxford: Clarendon Press.

Gödel, K., 2003b. *Collected Works, Vol. 5: Correspondence H–Z*. Oxford: Clarendon Press.

Goldrei, D., 1996. *Classic Set Theory*. Boca Raton, Florida: Chapman and Hall/CRC.

Goodstein, R. L., 1944. On the restricted ordinal theorem. *Journal of Symbolic Logic*, 9: 33–41.

Grandy, R. E., 1977. *Advanced Logic for Applications*. Dordrecht: D. Reidel.

Hájek, P. and Pudlák, P., 1993. *Metamathematics of First-Order Arithmetic*. Berlin: Springer.

Hale, B. and Wright, C., 2001. *The Reason's Proper Study*. Oxford: Clarendon Press.

Hart, W. D. (ed.), 1996. *The Philosophy of Mathematics*. Oxford Readings in Philosophy. Oxford University Press.

Heck, R., 2007. The logic of Frege's Theorem. Forthcoming in a *Festschrift* for Crispin Wright.

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

Hilbert, D., 1925. On the infinite. In van Heijenoort 1967, pp. 369–392.

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.

Hodges, W., 1997. *A Shorter Model Theory*. Cambridge: Cambridge University Press.

Hunter, G., 1971. *Metalogic*. London: Macmillan.

Isaacson, D., 1987. Arithmetical truth and hidden higher-order concepts. In The Paris Logic Group (ed.), *Logic Colloquium '85*. Amsterdam: North-Holland. Page references are to the reprint in Hart 1996.

Isles, D., 1992. What evidence is there that 2^{65536} is a natural number? *Notre Dame Journal of Formal Logic*, 33: 465–480.

Jeroslow, R. G., 1973. Redundancies in the Hilbert-Bernays derivability conditions for Gödel's second incompleteness theorem. *Journal of Symbolic Logic*, 38: 359–367.

Kalmár, L., 1959. An argument against the plausibility of Church's Thesis. In A. Heyting (ed.), *Proceedings of the Colloquium held at Amsterdam, 1957*, pp. 72–80. Amsterdam: North-Holland.

Kaye, R., 1991. *Models of Peano Arithmetic*. Oxford Logic Guides 15. Oxford: Clarendon Press.

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

Ketland, J., 1999. Deflationism and Tarski's paradise. *Mind*, 108: 69–94.

Ketland, J., 2005. Deflationism and the Gödel phenomena: reply to Tennant. *Mind*, 114: 75–88.

Kirby, L. and Paris, J., 1982. Accessible independence results for Peano arithmetic. *Bulletin of the London Mathematical Society*, 14: 285–293.

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., 1943. Recursive predicates and quantifiers. *Transactions of the American Mathematical Society*, 53: 41–73.

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.

Kleene, S. C., 1987. Reflections on Church's Thesis. *Notre Dame Journal of Formal Logic*, 28: 490–498.

Kolata, G., 1982. Does Gödel's theorem matter to mathematics? *Science*, 218: 779–780. Also in Harrington et al., 1985.

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

Kreisel, G., 1957. Abstract: A refinement of ω-consistency. *Journal of Symbolic Logic*, 22: 108–109.

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.

Löwenheim, L., 1915. On possibilities in the calculus of relatives. In van Heijenoort 1967, pp. 232–251.

Lucas, J. R., 1961. Minds, machines and Gödel. *Philosophy*, 36: 112–127.

Lucas, J. R., 1996. Minds, machines and Gödel: A retrospect. In P. J. R. Millican and A. Clark (eds.), *Machines and Thought: The Legacy of Alan Turing*. Oxford: Oxford University Press.

Mac Lane, S., 1986. *Mathematics: Form and Function*. New York: Springer-Verlag.

Mancosu, P. (ed.), 1998. *From Brouwer to Hilbert*. Oxford: Oxford University Press.

Martin, R. M., 1943. A homogeneous system for formal logic. *Journal of Symbolic Logic*, 8: 1–23.

Martin, R. M., 1949. A note on nominalism and recursive functions. *Journal of Symbolic Logic*, 14: 27–31.

Mendelson, E., 1964. *Introduction to Mathematical Logic*. Princeton, NJ: van Nostrand.

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*. Boca Raton, Florida: Chapman and Hall, 4th edn.

Meyer, A. R. and Ritchie, D., 1967. Computational complexity and program structure. Tech. Rep. RC-1817, IBM.

Milne, P., 2007. On Gödel sentences and what they say. Forthcoming in *Philosophia Mathematica*.

Moschovakis, Y., 2006. *Notes on Set Theory*. New York: Springer, 2nd edn.

Myhill, J., 1952. A derivation on number theory from ancestral theory. *Journal of Symbolic Logic*, 17: 192–197.

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

Odifreddi, P., 1999. *Classical Recursion Theory: The Theory of Functions and Sets of Natural Numbers*. Studies in logic and the foundations of mathematics, vol. 125. Amsterdam: North-Holland, 2nd edn.

Olszewski, A., Woleński, J., and Janusz, R. (eds.), 2006. *Church's Thesis after 70 Years*. Frankfurt: Ontos Verlag.

Parikh, R., 1971. Existence and feasibility in arithmetic. *Journal of Symbolic Logic*, 36: 494–508.

Paris, J. and Harrington, L. A., 1977. The incompleteness theorems. In J. Barwise (ed.), *Handbook of Mathematical Logic*, pp. 1133–1142. Amsterdam: North-Holland.

Parsons, C., 1980. Mathematical intuition. *Proceedings of the Aristotelian Society*, 80: 145–168.

Parsons, C., 1998. Finitism and intuitive knowledge. In M. Schirn (ed.), *The Philosophy of Mathematics Today*, chap. 249-270. Oxford University Press.

Peano, G., 1889. *The Principles of Arithmetic*. In van Heijenoort 1967, pp. 85–97.

Penrose, R., 1989. *The Emperor's New Mind*. Oxford: Oxford University Press.

Penrose, R., 1994. *Shadows of the Mind*. Oxford: Oxford University Press.

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*, 111: 42–60.

Péter, R., 1951. *Rekursive Funktionen*. Budapest: Akadémiai Kiadó. Translated as Péter 1967.

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

Pohlers, W., 1989. *Proof Theory*. Lecture Notes in Mathematics 1407. Springer-Verlag.

Potter, M., 2000. *Reason's Nearest Kin*. Oxford: Oxford University Press.

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

Presburger, M., 1930. Über die Vollständigkeit eines gewisse Systems der Arithmetik ganzer Zahlen, in welchem die Addition als einzige Operation hervortritt. In F. Leja (ed.), *Comptes-rendus du I Congrès des Mathématiciens des Pays Slaves, Varsovie 1929*, pp. 92–101. Translated as Presburger 1991.

Presburger, M., 1991. On the completeness of a certain system of arithmetic of whole numbers in which addition occurs as the only operation. *History and Philosophy of Logic*, 12: 225–233.

Pudlák, P., 1998. The length of proofs. In S. R. Buss (ed.), *Handbook of Proof Theory*, pp. 547–638. Amsterdam: Elsevier Science B.V.

Putnam, H., 1960. Minds and machines. In S. Hook (ed.), *Dimensions of Mind*, pp. 148–179. New York: New York University Press. In Putnam 1975, pp. 362–385.

Putnam, H., 1975. *Mind, Language and Reality: Philosophical Papers, Vol. 2*. Cambridge: Cambridge University Press.

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

Raatikainen, P., 2003. Hilbert's program revisited. *Synthese*, 137: 157–177.

Ramsey, F. P., 1925. The foundations of mathematics. *Proceedings of the London Mathematical Society*, 25: 338–384. Page references are to the reprint in Ramsey 1990.

Ramsey, F. P., 1990. *Philosophical Papers*. Cambridge: Cambridge University Press. Edited by D. H. Mellor.

Rautenberg, W., 2006. *A Concise Introduction to Mathematical Logic*. New York: Springer, 2nd edn.

Robinson, J., 1949. Definability and decision problems in arithmetic. *Journal of Symbolic Logic*, 14: 98–114.

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*, 1: 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., 1908. Mathematical logic as based on the theory of types. *American Journal of Mathematics*, 30: 222–262. Reprinted in Russell 1956.

Russell, B., 1956. *Logic and Knowledge: Essays 1901–1950*. 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.

Shapiro, S., 1991. *Foundations without Foundationalism: A Case for Second-Order Logic*. Oxford Logic Guides 17. Oxford: Clarendon Press.

Shapiro, S., 1998. Incompleteness, mechanism and optimism. *Bulletin of Symbolic Logic*, 4: 273–302.

Shapiro, S., 2006a. Computability, proof, and open-texture. In Olszewski et al. (2006), pp. 420–455.

Shapiro, S., 2006b. *Vagueness in Context*. Oxford: Clarendon Press.

Shen, A. and Vereshchagin, N. K., 2003. *Computable Functions*. Rhode Island, USA: American Mathematical Society.

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.

Simmons, K., 1993. *Universality and the Liar*. Cambridge: Cambridge University Press.

Simpson, S. G., 1991. *Subsystems of Second Order Arithmetic*. Berlin: Springer.

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.

Skolem, T., 1930. Über einige Satzfunktionen in der Arithmetik. *Skrifter utgitt av Det Norske Videnskaps-Akademi i Oslo, I. Mathematisk-naturvidenskapelig klasse*, 7: 1–28. Reprinted in Skolem 1970, pp. 281–306.

Skolem, T., 1970. *Selected Works in Logic*. Oslo: Universitetsforlaget.

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

Smoryński, C., 1982. The varieties of arboreal experience. *Mathematical Intelligencer*, 4: 182–189. Also in Harrington et al., 1985.

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

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

Smullyan, R. M., 1994. *Diagonalization and Self-Reference*. Oxford: Clarendon Press.

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

Szabo, M. E. (ed.), 1969. *The Collected Papers of Gerhard Gentzen*. Amsterdam: North-Holland.

Tait, W. W., 1981. Finitism. *Journal of Philosophy*, 78: 524–546. Reprinted in Tait 2005.

Tait, W. W., 2002. Remarks on finitism. In W. Sieg, R. Sommer, and C. Talcott (eds.), *Reflections on the Foundations of Mathematics: Essays in Honor of Solomon Feferman*, pp. 410–419. Association for Symbolic Logic and A K Peters. Reprinted in Tait 2005.

Tait, W. W., 2005. *The Provenance of Pure Reason. Essays in the Philosophy of Mathematics and Its History*. New York: Oxford University Press.

Takeuti, G., 1987. *Proof Theory*. Amsterdam: North-Holland.

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.

Tennant, N., 1978. *Natural Logic*. Edinburgh: Edinburgh University Press.

Tennant, N., 2002. Deflationism and the Gödel phenomena. *Mind*, 111: 551–582.

Tennant, N., 2005. Deflationism and the Gödel phenomena: reply to Ketland. *Mind*, 114: 89–96.

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.

Trakhtenbrot, B. A., 1988. Comparing the Church and Turing approaches: two prophetical messages. In R. Herken (ed.), *The Universal Turing Machine*, pp. 603–630. Oxford University Press.

Turing, A., 1936. On computable numbers, with an application to the *Entscheidungsproblem*. *Proceedings of the London Mathematical Society*, 42: 230–265. 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.

Visser, A., 1989. Peano's smart children: a provability logical study of systems with built-in consistency. *Notre Dame Journal of Formal Logic*, 30: 161–196.

von Neumann, J., 1927. Zur Hilbertschen Beweistheorie. *Mathematische Zeitschrift*, 26: 1–46.

Waismann, F., 1945. Verifiability. *Proceedings of the Aristotelian Society, Supplementary Volume*, 19: 119–150.

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

Wilkie, A. J. and Paris, J., 1987. On the scheme of induction for bounded arithmetic formulas. *Annals of Pure and Applied Logic*, 35: 261–302.

Willard, D. E., 2001. Self-verifying axiom systems, the incompleteness theorem, and related reflection principles. *Journal of Symbolic Logic*, 66: 536–596.

Wright, C., 1983. *Frege's Conception of Number as Objects*. Aberdeen: Aberdeen University Press.

Yaqub, A., 1993. *The Liar Speaks the Truth*. New York and Oxford: Oxford University Press.

Zach, R., 2003. Hilbert's program. In E. N. Zalta (ed.), *The Stanford Encyclopedia of Philosophy*. URL `http://plato.stanford.edu/archives/fall2003/entries/hilbert-program/`.

Zach, R., 2005. Review of Potter 2000. *Notre Dame Journal of Formal Logic*, 46: 503–513.

Zach, R., 2006. Hilbert's program then and now. In D. Jacquette (ed.), *Philosophy of Logic*. Amsterdam: North-Holland.

Index

Entries for *technical terms* and for *theorems* give the principal location(s) where you will find them explained. Entries for *names* link to not-merely-bibiliographical occurrences.