Gödel Without (Too Many) Tears

Second edition: iPad friendly version

Peter Smith

April 16, 2016

Contents

Preface **vi**

1 Incompleteness, the very idea **1**
Box: Kurt Gödel 1
1.1 The idea of an axiomatized formal theory 2
1.2 'Formally undecidable propositions' and negation incompleteness 6
1.3 Deductivism, logicism, and *Principia* 7
1.4 Gödel's bombshell 8
1.5 The First Incompleteness Theorem, more carefully 9
1.6 The First Incompleteness Theorem is better called an *incompletability* theorem 11
Box: Notational conventions 12
1.7 How did Gödel prove the First Theorem (in the semantic version)? 12
1.8 Gödel's completeness theorem vs his incompleteness theorem . . 15
Further reading 15

2 Incompleteness and undecidability **17**
2.1 Negation completeness and decidability 18
2.2 Capturing numerical properties in a theory 20
2.3 Sufficiently strong theories are undecidable 21
Box: Diagonalization 23
2.4 An incompleteness proof 23
Further reading 24

3 Two weak arithmetics **25**
3.1 Baby Arithmetic 26
3.1.1 The language L_B 26
3.1.2 The axioms and logic of BA 27

Box: Notation again . . . 28
3.2 Some proofs inside BA, and three little theorems . . . 29
3.3 BA is a sound and complete theory of the truths of L_B . . . 31
3.4 Robinson Arithmetic, Q . . . 32
3.4.1 The language L_A . . . 32
3.4.2 The axioms and logic of Q . . . 33
3.5 Robinson Arithmetic is not complete . . . 34
3.6 Statements of order in Robinson Arithmetic . . . 36
3.7 Why Q is interesting . . . 38
Further reading . . . 38

4 First-order Peano Arithmetic **40**
4.1 Arithmetical Induction . . . 42
4.1.1 The ω-rule . . . 42
4.1.2 Replacing an infinitary rule with a finite one . . . 43
4.1.3 Induction: the basic idea . . . 45
Box: A word to worried philosophers . . . 46
4.2 The induction axiom, the induction rule, the induction schema . . . 47
4.3 First-order Peano Arithmetic . . . 48
4.3.1 Getting generous with induction . . . 48
4.3.2 Introducing PA . . . 49
4.3.3 Summary overview of PA . . . 50
4.4 What PA can prove . . . 51
4.5 Burning questions . . . 52
Box: A compactness argument . . . 53
Further reading . . . 53

5 Quantifier complexity **55**
5.1 Q knows about bounded quantifications . . . 56
5.2 Δ_0, Σ_1 and Π_1 wffs . . . 56
Box: A remarkable corollary . . . 58
Further reading . . . 59

6 Primitive recursive functions **60**
6.1 Introducing the primitive recursive functions . . . 62
6.2 Defining the p.r. functions more carefully . . . 64
6.2.1 More on definition by primitive recursion . . . 64
6.2.2 The initial functions . . . 65
6.2.3 Primitive recursion and by composition: the official story 65

6.2.4 Putting everything together . . . 66
6.3 How to prove a result about all p.r. functions . . . 67
6.4 The p.r. functions are computable . . . 68
6.4.1 The basic argument . . . 68
6.4.2 Computing p.r. functions by 'for'-loops . . . 69
6.4.3 If you can compute it using 'for'-loops, it is p.r. . . . 70
6.5 Not all computable numerical functions are p.r. . . . 71
Box: Can we effectively list all computable numerical functions? . . . 73
6.6 Defining p.r. properties and relations . . . 74
Further reading . . . 75

7 Expressing and capturing the primitive recursive functions 76
7.1 L_A can express the factorial function . . . 78
7.1.1 What we need to express the factorial function . . . 78
7.1.2 Gödel's β-function . . . 79
7.1.3 Defining the factorial in L_A . . . 80
7.2 L_A can express all p.r. functions . . . 81
Box: The proof of Theorem 23 is constructive . . . 82
7.3 Canonical wffs for expressing p.r. functions are Σ_1 . . . 83
7.4 Q can capture all p.r. functions . . . 84
7.4.1 How to capture a function . . . 84
7.4.2 The proof strategy . . . 85
7.5 Expressing/capturing properties and relations . . . 86
Further reading . . . 87

8 The arithmetization of syntax 88
Box: Formalization and finitary reasoning . . . 90
8.1 Gödel-numbering . . . 94
8.1.1 Coding expressions in a formal language . . . 94
8.1.2 Coding sequences . . . 95
8.2 The arithmetization of syntactic properties/relations . . . 96
8.2.1 *Term*, *Wff* and *Sent* are p.r. properties . . . 96
8.2.2 *Prf* is a p.r. relation . . . 97
Box: Our results are robust! . . . 98
8.3 Some cute notation . . . 99
Further reading . . . 100

9 The First Incompleteness Theorem 101
9.1 The idea of diagonalization . . . 102

9.2 Constructing a Gödel sentence . . . 103
Box: What G says . . . 105
9.3 The First Theorem – the semantic version . . . 106
9.3.1 If PA is sound, it is incomplete . . . 106
9.3.2 Generalizing the proof . . . 106
9.3.3 Our Incompleteness Theorem is better called an *incompletability* theorem . . . 107
Box: Comparing old and new semantic incompleteness theorems . . . 108
9.4 ω-completeness, ω-consistency . . . 109
9.5 The First Theorem – the syntactic version . . . 111
9.5.1 If PA is consistent, it can't prove G . . . 111
9.5.2 If PA is consistent, it is ω-incomplete . . . 111
9.5.3 If PA is ω-consistent, it can't prove $\neg$G . . . 112
9.5.4 Putting together the syntactic Incompleteness Theorem for PA . . . 112
9.5.5 Generalizing the proof . . . 112
Box: Comparing old and new syntactic incompleteness theorems . . . 113
9.6 Gödel's own Theorem . . . 114
Further reading . . . 115

10 The Diagonalization Lemma, Rosser and Tarski 116
10.1 The Diagonalization Lemma . . . 118
10.2 Provability predicates . . . 119
10.3 Incompleteness from the Diagonalization Lemma . . . 120
Box: Proving our old G_T is in fact a fixed point for $\neg\mathsf{Prov}_T$ . . . 120
10.4 Tarski's Theorem . . . 122
10.4.1 Truth-predicates and truth-definitions . . . 122
10.4.2 The undefinability of truth . . . 123
10.4.3 The inexpressibility of truth . . . 124
Box: The Master Argument for incompleteness . . . 124
10.5 Rosser's Theorem . . . 125
10.5.1 Rosser's basic trick . . . 125
10.5.2 Implementing the trick . . . 126
10.5.3 Rosser's Theorem . . . 126
Further reading . . . 127

11 The Second Incompleteness Theorem 128
11.1 Defining Con_T . . . 129
11.2 From the Formalized First Theorem to the Second Theorem . . . 130

Box: Historical note 131
11.3 How interesting is the Second Theorem? 131
11.4 Sharpening the Second Theorem 132
11.5 The box notation 133
11.6 Proving the Formalized First Theorem 133
11.7 Other excitements 135
11.7.1 G_T and Con_T are provably equivalent in T 135
11.7.2 Con is a fixed point of the provability predicate 136
Box: Theories that 'prove' their own inconsistency 136
Further reading 137

12 Curry's Paradox and Löb's Theorem **139**
12.1 Curry's Paradox: How to show that the moon is made of green cheese 140
12.2 Curry's theorem, and two implications 141
12.3 Proving the moon is made of green cheese again and Löb's Theorem 143
12.4 The Second Incompleteness Theorem again 147
Box: What's still to come 147

Preface

Why these notes? After all, I've already written a long book, *An Introduction to Gödel's Theorems*. Surely that's more than enough to be going on with?

Ah, but there's the snag. It *is* more than enough. In the writing, as is the way with these things, the book grew far beyond the scope of the lecture notes from which it started. And while I hope the result is still very accessible to someone prepared to put in the required time and effort, there is – to be frank – a *lot* more in the book than is really needed by philosophers meeting the incompleteness theorems for the first time, or indeed even by mathematicians wanting a brisk introduction.

Some might therefore appreciate something like a cut-down version of the book – an introduction to the *Introduction*. Hence *Gödel Without (Too Many) Tears*.

An earlier edition – call it *GWT1* – was written for the last couple of outings of a short lecture course I had given in Cambridge for a number of years (which was also repeated at the University of Canterbury, NZ). Many thanks to generations of students for useful feedback.

The notes were intended, then, to help bridge the gap between relaxed classroom talk'n'chalk which just highlighted the Really Big Ideas, and the more detailed treatments of topics in my book. However, despite that intended role, I did try to make *GWT1* reasonably stand-alone. And since publishing it on my website, it has seems to have been treated as a welcome resource (being downloaded nearly three thousand times in 2013 alone).

That earlier edition of these notes was tied, of course, to the first edition of my book, *IGT1*, as published by CUP in 2007 (with some later revised reprints). A significantly improved second edition of the book *IGT2* was published in 2013, which should by now be widely available in libraries (and as a relatively cheap paperback). It is time to update *GWT1* to make a corresponding second edition

of these notes, and so here is a complete draft of *GWT2* (now itself in a revised version).

If you have come across these notes from some other source than my website – quite possible because copies inevitably escape into the wild! – then do check that you've got the latest version by visiting http://www.logicmatters.net/igt. Compare the date on the cover page of this version of the notes with the date of the latest version as announced on the website.

A number of people have kindly let me know about typos in the first release of this edition, which have now been corrected. More importantly, Henning Makholm sent very useful substantive comments, and I have tried to meet his critical points in this revision.

If you notice any more mistakes, typos or thinkos, in *GWT2* and/or *IGT2* please let me know by email to peter_smith@me.com. If you want to raise a more substantive query which might involve some issue of wider interest, why not ask a question at that wonderful resource http://math.stackexchange.com?

Episode 1

Incompleteness, the very idea

- Kurt Gödel, very briefly
- The idea of an (effectively) axiomatized formal theory
- Undecidable propositions, negation-incompleteness
- What *Principia* is about
- Gödel's bombshell: the First Incompleteness Theorem
- Why incompleteness can't easily be repaired
- How Gödel proved (one version of) his First Theorem
- A reality check: why there's no tension between Gödel's completeness theorem and his incompleteness result

Kurt Gödel (1906–1978) was by common consent the greatest logician of the twentieth century. Born in what is now Brno, and educated in Vienna, Gödel left Austria for the USA in 1938, and spent the rest of his life at the Institute of Advanced Studies at Princeton.

Gödel's doctoral dissertation, written when he was 23, established the *completeness* theorem for the first-order predicate calculus (i.e. a standard

proof system for first-order logic indeed captures all the semantically valid inferences).

Later he would do immensely important work on set theory, as well as make seminal contributions to proof theory and to the philosophy of mathematics. He even wrote on models of General Relativity with 'closed timeline curves' (where, in some sense, time travel is possible). Always a perfectionist, after the mid 1940s he more or less stopped publishing.

Talk of 'Gödel's Theorems' typically refers, however, to his two *incompleteness* theorems in an epoch-making 1931 paper. And it is these theorems, and more particularly, the First Incompleteness Theorem, that these notes are all about. (Yes, that's right: Gödel proved a 'completeness theorem' and also an 'incompleteness theorem'. By the end of this first episode you should understand the difference!)

Putting it crudely, and a little bit tendentiously, the incompleteness theorems show that the ambitions of two major foundational programs – logicism (about which a bit more in this episode) and Hilbert's program (about which more later) – can't be achieved. The consequent impact of the theorems on logic and foundational studies is hard to exaggerate.

1.1 The idea of an axiomatized formal theory

The title of the 1931 paper which proves the First Incompleteness Theorem is **'On formally undecidable propositions of *Principia Mathematica* and related systems I'**.

The 'I' here indicates that it is the first Part of what was going to be a two part paper, with Part II spelling out the proof of the Second Theorem which is only very briefly indicated in Part I. But Part II was never written. We'll see much later why Gödel thought he didn't need to bother.

This title itself gives us a number of things to explain. What's a 'formally undecidable proposition'? What's *Principia Mathematica*? – well, you've quite probably heard of that triple-decker work by A. N. Whitehead and Bertrand Russell, now more than a century old and little read: but what is the project of that book? What counts as a 'related system' – related, that is, to the one in the book? In fact, just what is meant by 'system' here?

We'll take the last question first. A 'system' (in the relevant sense) is an **(effectively) axiomatized formal theory**. What does that mean? Well, the

headline news is that a theory T counts as such a theory just in case it has

(i) an effectively formalized language L,

(ii) an effectively decidable set of axioms,

(iii) an effectively formalized proof-system in which we can deduce theorems from the axioms.

But to explain what we mean here, we first need to take some steps towards pinning down the intuitive notion of **effective decidability**.

So let's say:

Defn. 1. *A property P (defined over some domain of objects D) is effectively decidable iff there's an algorithm (a finite set of instructions for a deterministic computation) for settling in a finite number of steps, for any $o \in D$, whether o has property P. In other words, there's a step-by-step mechanical routine for settling the issue, so a suitably programmed deterministic computer could in principle do the trick.*

Likewise, a set Σ is effectively decidable if the property of being a member of that set is effectively decidable.

Is that satisfactory, though? We've invoked the idea of what a computer (in principle) could do by implementing some algorithm. And doesn't that leave quite a bit of slack in the definition? Why shouldn't what a computer can do depend, for example, on its architecture (even if we idealize, and e.g. put no time limit on its computations)?

Well, it turns out that in fact the notion of effective decidability is very robust: what is algorithmically computable-in-principle according to any given sensible sharpened-up definition turns out to be exactly what is algorithmically computable-in-principle by any other sensible sharpened-up definition. Of course, it's not at all trivial that this is how things are going to pan out. So just for the moment you are going to have to take it on trust (sorry!) that the idea of computability-in-principle can in fact be given nice sharp definitions that all come to same thing.

Assuming then that Defn. 1 *can* be suitably sharpened to make the idea of effective decidability in good shape, now take in turn those conditions (i) to (iii) for being a formal axiomatized theory.

(i) We'll assume that the basic idea of a **formal language** L is familiar from earlier logic courses. Though there's an issue about terminology we'd better clarify.

A language, for us, is a bunch of symbols with an intended interpretation: so a language, for us, has a **syntax** *and* a **semantics**. The syntax fixes which strings of symbols form terms, which form wffs, and in particular which strings of symbols form *sentences*, i.e. closed wffs with no unbound variables dangling free. The semantics assigns truth-conditions to each sentence of the language.

It is not at all unusual for logic-books to talk of a language when speaking just of uninterpreted strings of symbols. But I think we should deprecate that usage. Sometimes below I'll talk about an 'interpreted language' for emphasis: but strictly speaking – in my idiolect – that's redundant.

Now, to emphasize what is perhaps not emphasized in introductory courses,

Defn. 2. *We will take an effectively formalized language L to have a finite stock of basic symbols, and then, crucially, the syntactic rules of L must be such that the properties of being a term, a wff, a wff with one free variable, a sentence, etc., are effectively decidable.*

Actually, we don't strictly need to require the stock of basic symbols to be finite so long as it is decidable what's a symbol, but we lose no useful generality by assuming finitude here. For the usual way of ensuring a formal language is indeed effectively formalized is of course to specify some finite basic vocabulary, and then give rules for building up expressions from this basic vocabulary in such a way that we can mechanically decide whether an expression is indeed constructible by the rules. And why do we want the properties of being a sentence, etc., to be effective decidable? Well, the whole point of setting up a formal language is, for a start, precisely to put issues of what is and isn't a sentence beyond dispute, and the best way of doing that is to ensure that even a suitably programmed computer could decide whether a string of symbols is or is not a sentence of the language.

A formal language will also normally have an intended *semantics* which gives the interpretation of L, fixing truth conditions for each L-sentence. Again, the semantics should be presented in such a way that we can mechanically read off from the interpretation rules the interpretation of any given sentence. Moreover, we normally require that any sentence gets a *unique* interpretation. That's because another key thing we want in setting up a formal language is to avoid any ambiguity. However, it is the syntactic features of formalized languages that are going to be crucial for us later, so that's why we highlight just them in our Defn. 2.

(ii) Some logic books define a **theory** as just any set of sentences. But it is better to say that a theory is a set of sentences closed under a consequence relation. In

other words, a theory comes with some deductive apparatus, and any sentence that can be deduced from other sentences in the theory using that apparatus is also part of the theory. Then a bunch of sentences Σ counts as giving **axioms** for the theory T if every sentence in T can be deduced from Σ using the deductive apparatus.[1]

We will require a properly formalized theory T of this kind, with a set of axioms picked out, to be such that it is effectively decidable what's an axiom. For if we are in the business of pinning down a theory by axiomatizing it, then we will normally want to avoid any possible dispute about what counts as an axiom by ensuring that we can mechanically decide whether a given sentence is indeed one of the axioms.

(iii) Just laying down a bunch of axioms would normally be pretty idle if we can't deduce conclusions from them! So a formal axiomatized theory T of course normally comes equipped with a deductive **proof-system**, a set of rules for deducing further theorems from our initial axioms. But a proof-system such that we couldn't routinely tell whether its rules are in fact being followed again wouldn't have much point for practical purposes. Hence we naturally also require that it is effectively decidable whether a given array of wffs is indeed a proof from the axioms according to the rules. It doesn't matter for our purposes whether the proof-system is e.g. a Frege/Hilbert axiomatic logic, a natural deduction system, a tree/tableau system – so long as it is indeed effectively checkable that a candidate proof-array has the property of being properly constructed according to the rules.

So, in summary of (i) to (iii),

Defn. 3. *An (effectively) axiomatized formal theory T has an effectively formalized language L, a certain class of L-sentences picked out as axioms where it is effectively decidable what's an axiom, and it has a proof-system such that it is effectively decidable whether a given array of wffs is indeed a proof from the axioms according to the rules.*

Careful, though! To say that, for a properly formalized theory T it must be effectively decidable whether a given purported T-proof of φ is indeed a kosher proof according to T's deduction system is not, repeat *not*, to say that it must be effectively decidable whether φ has a proof. It is one thing to be able to effectively *check* a proof once proposed, it is another thing to be able to effectively *decide*

[1] We'll allow the extreme case where the set of axioms Σ is null. It should be familiar that we can trade in axioms for rules of inference – though we can't trade in *all* rules of inference for axioms if we want to be able to deduce anything: cf. Lewis Carroll's Achilles and the Tortoise!

in advance whether there exists a proof to be discovered. (It will turn out, for example, that any formal axiomatized theory T containing a certain modicum of arithmetic is such that, although you can mechanically check a purported proof of φ to see whether it *is* a proof, there's no mechanical way of telling of an arbitrary φ whether it is provable in T or not.)

1.2 'Formally undecidable propositions' and negation incompleteness

Henceforth when we talk about formal theories or axiomatized formal theories, we always mean effectively axiomatized formal theories (unless we explicitly say otherwise).

Some absolutely standard logical notation, applied to formal theories:

Defn. 4. *'$T \vdash \varphi$' says: there is a formal deduction in T's proof-system from T-axioms to the formula φ as conclusion. If φ is a sentence and $T \vdash \varphi$, then φ is said to be a* theorem *of T.*

Compare:

Defn. 5. *'$T \vDash \varphi$' says: any model (re)interpreting the non-logical vocabulary that makes all the axioms of T true makes φ true.*

So '$\vdash$' officially signifies **provability** in T, a formal syntactically definable relation, while '$\vDash$' signifies **semantic entailment**, where '$T \vDash \varphi$ is true if any model which makes the axioms of T true makes φ true too.

Of course, we will require that a deductive system respects semantic entailment so that $T \vdash \varphi$ only if $T \vDash \varphi$. In a word, we require a deductive system in a sensible theory to be **sound** (with respect to the intended semantics). We can't always insist on the converse, however. Still, in the special case where T is built in a classical first-order language, and T's deductive proof-system is a standard first-order logic, we *will* have that if $T \vDash \varphi$, then $T \vdash \varphi$. For standard first-order deductive systems are **complete**. That was first proved for a Hilbert-style deductive system by Gödel in his doctoral thesis: hence **Gödel's completeness theorem**.

Defn. 6. *If T is a theory, and φ is some sentence of the language of that theory, then T* **formally decides** *φ iff either $T \vdash \varphi$ or $T \vdash \neg\varphi$.*

Hence,

Defn. 7. *A sentence φ is* **formally undecidable** *by T iff $T \nvdash \varphi$ and $T \nvdash \neg\varphi$.*

Another bit of terminology:

Defn. 8. *A theory T is* **negation complete** *iff it formally decides every closed wff of its language – i.e. for every sentence φ, $T \vdash \varphi$ or $T \vdash \neg\varphi$*

Trivially, then, there are 'formally undecidable propositions' in T if and only if T isn't negation complete.

Of course, it is very, very easy to construct negation-incomplete theories: just leave out some necessary basic assumptions about the matter in hand!

But suppose we are trying to fully pin down some body of truths using a formal theory T. We fix on an interpreted formal language L apt for expressing such truths. And then we'd ideally like to lay down some axioms framed in L such that when (but only when) φ is true, $T \vdash \varphi$. So, making the classical assumption that either φ is true or $\neg\varphi$ is true, we'd very much like T to be such that either $T \vdash \varphi$ or $T \vdash \neg\varphi$ but not both. Negation completeness, then, is a natural desideratum for theories.

1.3 Deductivism, logicism, and *Principia*

The elementary arithmetic of successor ('next number'), addition, and multiplication is child's play (literally!). It is entirely plausible to suppose that, whether the answers are readily available to us or not, questions posed in what we'll call **the language of basic arithmetic**[2] have entirely determinate answers. Here 'the language of basic arithmetic' means a language which has terms for the natural numbers (for zero and its successors – so having a term for zero and a sign for the successor function is enough), plus function signs for addition and multiplication, plus the usual first order logical apparatus.

The answers to questions posed in this language of basic arithmetic are surely 'fixed' by (a) the fundamental zero-and-its-successors structure of the natural number series (with zero not being a successor, every number having a successor, distinct numbers having distinct successors, and so the sequence of zero and its successors never circling round but marching off for ever) plus (b) the nature of addition and multiplication as given by the school-room explanations.

So it is surely plausible to suppose that we should be able lay down a bunch of axioms which characterize the number series, addition and multiplication (which

[2]The interpreted language, of course – or no questions are posed!

codify what we teach the kids), and that these axioms should settle every truth of basic arithmetic, in the sense that every such truth of the language of successor, addition, and multiplication is logically provable from these axioms. For want of a standard label, call this view **deductivism** about basic arithmetic.

What could be the status of the axioms? I suppose you might, for example, be a Kantian deductivist who holds that the axioms encapsulate 'intuitions' in which we grasp the fundamental structure of the numbers and the nature of addition and multiplication, where these 'intuitions' are a special cognitive achievement in which we somehow represent to ourselves the arithmetical world.

But talk of such intuition is very puzzling and problematic. So we could very well be tempted instead by Frege's seemingly more straightforward view that the axioms are **analytic**, themselves truths of logic-plus-definitions. On this view, we don't need Kantian 'intuitions' going beyond logic: logical reasoning from definitions is enough to get us the axioms of arithmetic. The Fregean brand of deductivism is standardly dubbed **'logicism'**.

Now, famously, Frege's attempt to be a logicist deductivist about basic arithmetic (in fact, for him, about more than basic arithmetic) hit the rocks, because – as Russell showed – his logical system is in fact inconsistent in a pretty elementary way (it is beset by Russell's Paradox). That devastated Frege, but Russell was undaunted, and still gripped by deductivist ambitions he wrote:

> All mathematics [yes! – *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.

That's a huge promissory note in Russell's *The Principles of Mathematics* (1903). And *Principia Mathematica* (three volumes, though unfinished, 1910, 1912, 1913) is Russell's attempt with Whitehead to start making good on that promise. The project is to set down some logical axioms and definitions and deduce the laws of basic arithmetic from them (and then a lot more). Famously, the authors eventually get to prove that $1+1=2$ at *110.643 (Volume II, page 86), accompanied by the wry comment, 'The above proposition is occasionally useful'.

1.4 Gödel's bombshell

Principia, frankly, is a bit of a mess – in terms of clarity and rigour, it's quite a step backwards from Frege. And there are technical complications which mean that not all *Principia*'s axioms are clearly 'logical' even in a stretched sense. In

particular, there's an appeal to a brute-force *Axiom of Infinity* which in effect states that there is an infinite number of objects; and then there is the notoriously dodgy *Axiom of Reducibility*.[3] But leave those worries aside – they pale into insignificance compared with the bomb exploded by Gödel.

For Gödel's First Incompleteness Theorem sabotages not just the grandiose project of *Principia*, but shows that any form of deductivism about even just basic arithmetic (and didn't that, at least, seem an attractive idea?) is in fatal trouble.

Why? Well the proponent of deductivism about basic arithmetic (logicist or otherwise) wants to pin down first-order arithmetical truths about successor/addition/multiplication, without leaving any out: so he wants to give a negation-complete theory. *And there can't be such a theory.* Gödel's First Theorem says – at a very rough first shot – that **nice theories containing enough basic arithmetic are always negation incomplete**.

So varieties of deductivism, and logicism in particular, must always fail. Which is a rather stunning result![4]

1.5 The First Incompleteness Theorem, more carefully

Three more definitions. First, let's be a bit more careful about that idea of 'the language of basic arithmetic':

Defn. 9. *The formalized interpreted language L* contains the language of basic arithmetic *if L has at least the standard first-order logical apparatus (including identity), has a term '*0*' which denotes zero and function symbols for the successor, addition and multiplication functions defined over numbers – either built-in*

[3] *Principia* without the dodgy Axiom (and without the Axiom of Infinity) is a so-called 'type theory' which is quite nicely motivated, but you can't reconstruct much maths in it: the dodgy Axiom of Reducibility allows you to reconstruct classical maths by pretending that the type distinctions by which we are supposed to avoid paradox can be ignored when we need to do so.

[4] 'Hold on! I've heard of neo-logicism which has its enthusiastic advocates. How can that be so if Gödel showed that logicism is a dead duck?' Well, we might still like the idea that some logical principles plus what are more-or-less definitions (in a language richer than that of first-order logic) together *semantically* entail all arithmetical truths, while allowing that we can't capture the relevant entailment relation in a single properly axiomatized deductive system of logic. Then the resulting overall system of arithmetic won't count as a formal axiomatized theory of all arithmetical truth since its logic is not formalizable, and Gödel's theorems don't apply. But all that is another story, though we'll be touching on relevant issues later.

as primitives or introduced by definition – and has a predicate whose extension is the natural numbers.

Fine print: The point of that last clause is that if 'N' is a predicate satisfied just by numbers, then the wff $\forall \mathsf{x}(\mathsf{N}\mathsf{x} \rightarrow \varphi(\mathsf{x}))$ says that every number satisfies φ; so L can make general claims specifically about natural numbers. (If L is already defined to be a language whose quantifiers run over the numbers, then you could use '$\mathsf{x} = \mathsf{x}$' for 'N', or – equivalently – just forget about it!)

Defn. 10. *A theory T is* **sound** *iff its axioms are true (on the interpretation built into T's language), and its logic is truth-preserving, so all its theorems are true.*

Defn. 11. *A theory T is* **consistent** *iff there is no φ such that $T \vdash \varphi$ and $T \vdash \neg\varphi$,*

where '$\neg$' is T's negation operator (in a classical setting, then, if T is inconsistent, then $T \vdash \psi$ for all ψ). And of course, trivially, soundness implies consistency.

Gödel now proves (more accurately, gives us most of the materials to prove) the following:

Theorem 1. *If T is a sound formal axiomatized theory whose language contains the language of basic arithmetic, then there will be a true sentence G_T of basic arithmetic such that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$, so T is negation incomplete.*

However that *isn't* what is usually referred to as the First Incompleteness Theorem. For note, Theorem 1 tells us what follows from a **semantic** assumption, namely that T is sound. And soundness is defined in terms of truth. Now, post-Tarski, we aren't particularly scared of the notion of the truth. To be sure, there are issues about how best to treat the notion formally, to preserve as many as possible of our pre-formal intuitions while e.g. blocking versions of the Liar Paradox. But most of us think that we don't have to regard the general idea of truth as *metaphysically* loaded in an obscure and worrying way. But Gödel was writing at a time when, for various reasons (think logical positivism!), the very idea of truth-in-mathematics was under some suspicion. So it was *extremely* important to Gödel to show that you don't need to deploy any semantic notions to get (again roughly) the following result that only invokes the **syntactic** assumption of consistency (and another syntactic assumption).

Theorem 2. *For any consistent formal axiomatized theory T which can prove a certain modest amount of arithmetic (and has a certain additional desirable syntactically definable property that any sensible formalized arithmetic will share),*

there is a sentence of basic arithmetic G_T *such that* $T \nvdash \mathsf{G}_T$ *and* $T \nvdash \neg\mathsf{G}_T$*, so* T *is negation incomplete.*

Of course, we'll need to be a lot more explicit in due course, but that indicates the general character of Gödel's result. And the 'can prove a modest amount of arithmetic' is what makes a theory sufficiently related to *Principia*'s for the theorem to apply – remember the title of Gödel's paper! I'll not pause in this first episode to spell out just how much arithmetic that is, but we'll find that it is stunningly little. (Nor will I pause now to explain that 'additional desirable property' condition. We'll meet it in due course, but also explain how – by a cunning trick discovered by J. Barkley Rosser in 1936 – we can drop that condition.)

Note, to repeat, that the incompleteness theorem here comes in two flavours. There's a version making a semantic assumption (plus a weak assumption about what T can express), and there's a version making just a syntactic assumption (plus a stronger assumption about what T can prove). It is important to keep in mind that there are these two flavours of the First Theorem when reading other presentations.

1.6 The First Incompleteness Theorem is better called an *incompletability* theorem

Let's concentrate on the first, semantic, version of the First Incompleteness Theorem.

Suppose T is a sound theory which contains the language of basic arithmetic. Then we can find a true G_T such that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$. *Of course, that* doesn't *mean that* G_T *is 'absolutely unprovable', whatever that could mean. It just means that* G_T*-is-unprovable-in-*T.

Ok, why don't we just 'repair the gap' in T by adding G_T as a new axiom? So consider the theory $U = T + \mathsf{G}_T$ (to use an obvious notation). Then (i) U is still sound (for the old T-axioms are true, the added new axiom is true, and the logic is still truth-preserving). (ii) U is still a properly formalized theory, since adding a single specified axiom to T doesn't make it undecidable what is an axiom of the augmented theory. (iii) U contains the language of basic arithmetic. So Gödel's First Incompleteness Theorem applies, and we can find a sentence G_U such that $U \nvdash \mathsf{G}_U$ and $U \nvdash \neg\mathsf{G}_U$. And since U is stronger than T, we have a fortiori, $T \nvdash \mathsf{G}_U$ and $T \nvdash \neg\mathsf{G}_U$. In other words, 'repairing the gap' in T by adding G_T as a new

axiom leaves some other sentences that are undecidable in T *still* undecidable in the augmented theory.

And so it goes. Keep chucking more and more additional true axioms at T and our theory still remains negation-incomplete, unless it stops being sound or stops being effectively axiomatizable. In a good sense, T is **incompletable**.

Before going on, we should highlight a couple of useful **notational conventions** that we'll be using from now on in these notes (the same convention is used in *IGT2*, and indeed is, in some version or other, quite a common one):

1. Particular expressions from formal systems – and abbreviations of them – will be in sans serif type. Examples: $\mathsf{SS0} + \mathsf{S0} = \mathsf{SSS0}$, $\forall \mathsf{x} \forall \mathsf{y}(\mathsf{x} + \mathsf{y} = \mathsf{y} + \mathsf{x})$.

2. Expressions in informal mathematics will be in ordinary serif font (with variables, function letters etc. in italics). Examples: $2+1 = 3$, $n+m = m+n$, $S(x+y) = x+Sy$.

3. Greek letters, like the 'φ' we've just been using, are schematic variables in the metalanguage in which we talk about our formal systems.

For more explanations, see *IGT2*, §5.1.

1.7 How did Gödel prove the First Theorem (in the semantic version)?

And what about the *Second* Incompleteness Theorem that Gödel states (but doesn't prove) in his 1931 paper? What does that assert?

Good question – and the opening chapter of *IGT2* will in fact tell you a little about it. But here in these notes we'll maintain the suspense, and we will finish this first episode by instead outlining how Gödel proved the semantic version of his *First* Incompleteness Theorem. Obviously we'll be coming back to this in a lot more detail later, but some readers might find these outline hints very intriguing and an incentive to keep going. (Don't worry, however, if you find them too mystifying: we'll be explaining it all in slow motion later.)

We kick off with two natural definitions.

Defn. 12. *If L contains the language of basic arithmetic, so it contains a term* $\mathsf{0}$ *for zero and a function expression* S *for the successor function, then it contains the terms* $\mathsf{0}$, $\mathsf{S0}$, $\mathsf{SS0}$, $\mathsf{SSS0}$, *..., and these are L's* **standard numerals**. *We'll use '$\overline{\mathsf{n}}$' to abbreviate the standard numeral for n.*

(The overlining convention to indicate standard numerals is itself a standard one!) And henceforth, we'll assume that the language of any theory we are interested in contains the language of basic arithmetic and hence includes standard numerals denoting the numbers.

Defn. 13. *The formal wff $\varphi(\mathsf{x})$ of the interpreted language L* **expresses** *the numerical property P iff $\varphi(\overline{\mathsf{n}})$ is true on interpretation just when n has property P. Similarly, the formal wff $\psi(\mathsf{x},\mathsf{y})$ expresses the numerical relation R iff $\psi(\overline{\mathsf{m}},\overline{\mathsf{n}})$ is true just when m has relation R to n. And the formal wff $\chi(\mathsf{x},\mathsf{y})$ expresses the numerical function f iff $\chi(\overline{\mathsf{m}},\overline{\mathsf{n}})$ is true just when $f(m) = n$.*

The generalization to many-place relations/many-argument functions is obvious.

Then Gödel's proof of Theorem 1 in outline form goes as follows:

1. *Set up a Gödel numbering* We are nowadays familiar with the idea that all kinds of data can be coded up using numbers. So suppose we set up a sensible (effective) way of coding wffs and sequences of wffs by natural numbers – so-called **Gödel-numbering**. Then, given an effectively axiomatized formal theory T, we can define e.g. the numerical properties Wff_T, Sent_T and Prf_T where

   > $\mathit{Wff}_T(n)$ iff n is the code number of a T-wff.
   > $\mathit{Sent}_T(n)$ iff n is the code number of a T-sentence.
   > $\mathit{Prf}_T(m,n)$ iff m is the code number of a T-proof of the T-sentence with code number n.

   Since T is an effectively axiomatized theory, these numerical properties/relations are effectively decidable properties of numbers/relations. Why? Well remember it is (for example) effectively decidable whether a supposed T-proof-array is the genuine article and proves its purported conclusion. So it is effectively decidable whether the array which gets the code number m is indeed a T-proof of the conclusion coded by n. That is to say, it is effectively decidable whether $\mathit{Prf}_T(m,n)$. Similarly for the the properties Wff_T and Sent_T.

2. *Expressing such properties/relations inside T* We next show that such properties/relations can be expressed inside T by wffs of the formal theory belonging to the language of basic arithmetic [this takes quite a bit of work to show!]. We show in particular how to build – just out of the materials of the language of basic arithmetic – an arithmetic formal wff we'll abbreviate $\mathsf{Prf}_T(\mathsf{x}, \mathsf{y})$ that expresses the property Prf_T, so $\mathsf{Prf}_T(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ is true exactly when $Prf_T(m, n)$, i.e. when m is the code number of a T-proof of the wff with number n. What we rely on (to a first approximation) is that the language of basic arithmetic turns out to be *really* good at expressing decidable properties and relations, and relations like $Prf_T(m, n)$ are decidable because T is a formalized theory.

   Or rather (to a better approximation) we rely on the fact that basic arithmetic is very good at expressing so-called primitive recursive relations and for sensible theories $Prf_T(m, n)$ is primitive recursive – so actually, the argument doesn't depend on the informal notion of effective decidability.

3. *The construction: building a Gödel sentence* Now put $\mathsf{Prov}_T(\overline{\mathsf{n}}) =_{\text{def}} \exists \mathsf{x} \mathsf{Prf}_T(\mathsf{x}, \overline{\mathsf{n}})$ (where the quantifier is restricted, if necessary, to run over numbers). This says that some number numbers a proof of the wff with Gödel-number n; i.e. the wff with number n is a theorem.

   Next – the really cunning bit, but surprisingly easy – we show how to use $\mathsf{Prov}_T(\overline{\mathsf{n}})$ to build a **'Gödel' sentence** G_T whose Gödel number is g, in such a way that G_T is in fact logically equivalent to $\neg\mathsf{Prov}_T(\overline{\mathsf{g}})$. So G_T is true on interpretation iff $\neg\mathsf{Prov}_T(\overline{\mathsf{g}})$ is true, i.e. iff the wff number g is not a theorem, i.e. iff G_T is not a theorem.

   In sum, G_T is true if and only if it isn't a theorem. (Indeed, stretching a point, we might pretend that G_T says 'I am unprovable in T'.)

4. *The argument* It's now plain sailing! Assume T is sound. Suppose $T \vdash \mathsf{G}_T$. Then G_T would be a theorem, and hence G_T would be false, so T would have a false theorem and hence not be sound, contrary to the hypothesis that T is sound. So $T \nvdash \mathsf{G}_T$. So G_T is true. So $\neg\mathsf{G}_T$ is false and T, being sound, can't prove it. Hence we also have $T \nvdash \neg\mathsf{G}_T$.

There are big gaps to fill there, but that's the overall strategy for proving Theorem 1. If T is a sound theory whose language contains the language of basic arithmetic, then we can construct along the lines sketched a true sentence G_T of basic arithmetic such that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$, so T is negation incomplete.

The proof of Theorem 2 then uses the same construction of a Gödel sentence in proving incompleteness, but now time trading in the semantic assumption that

T is sound for the syntactic assumption that T is consistent (we also require a bit more, and require T to have that currently mysterious 'additional desirable [syntactic] property'). More about this in due course.

Of course, you might immediately think that there is something a bit worrying about our sketch. For basically, I'm saying we can construct an arithmetic sentence in T that, via the Gödel number coding, says 'I am not provable in T'. But shouldn't we be suspicious about that? After all, we know we get into paradox if we try to play with a **Liar sentence** that says 'I am not true'. So why does the self-reference in the Liar sentence lead to *paradox*, while the self-reference in Gödel's proof give us a *theorem*? A very good question indeed. I hope that over the coming episodes, the answer to that good question will become clear!

1.8 Gödel's completeness theorem vs his incompleteness theorem

One thing, though, we can get clear straight away. The old **completeness theorem**, that you are familiar with from intro. logic courses, is specifically about the relation between semantics and syntax in first-order systems and comes to the following:

> If T is a theory cast in a first-order language with a standard first-order deductive apparatus, then for any φ, if $T \vDash \varphi$ then $T \vdash \varphi$.

The deductive logic is complete because every conclusion that semantically follows from some given premisses can be deduced from them by formal logical proof. Compare (one version of) the **incompleteness theorem**.

> If T is a sound theory that can express enough arithmetic, then T is not negation-complete, i.e. there is some φ such that neither $T \vdash \varphi$ nor $T \vdash \neg\varphi$.

There is no evident tension at all between these two quite different results: think about it! And we'll soon be stating some first-order theories with standard (and hence complete) deductive proof-systems which are negation incomplete theories of arithmetic.

Further reading

At this point read *IGT2*, §§1.1–1.7, 3.1, 3.2, 4.1–4.4, 5.1. And for parallel reading see e.g. Richard E. Hodel, *An Introduction to Mathematical Logic* (Dover Publications reprint, 2013), §§1.1–1.7: you might as well include §1.4 on sets, functions and relations, and §1.5 on countable and uncountable sets since we'll need an amount of that material too very shortly.

§4.1 of *IGT2* on 'Formalization as an ideal' is particularly important if you haven't really come across the idea of a formalized theory before. For you do need to grasp 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 the lack of inferential gaps'.

If you want a little more about the informal notion of an algorithm which we appealed to in defining effective decidability, then you probably can't do much better than look at §1.1 of an absolute classic, Hartley Rogers, *Theory of Recursive Functions and Effective Computbaility* (McGraw-Hall, 1967): the section is indeed called 'The informal notion of algorithm'. There's a lot more about the notion available at http://en.wikipedia.org/wiki/Algorithm.

For an overview of Gödel's life and work, see http://en.wikipedia.org/wiki/Kurt_Gödel, or better – though you'll need to skip – http://plato.stanford.edu/entries/goedel. There is also a rather good and accessible full biography by John Dawson, *Logical Dilemmas* (A. K. Peters, 1997), which is very well worth reading sometime as it will also give you a real sense of the logical scene in the glory days of the 1930s.

For more on Russell's Paradox, see http://plato.stanford.edu/entries/russell-paradox/. And for more on *Principia* see http://plato.stanford.edu/entries/principia-mathematica/.

Episode 2

Incompleteness and undecidability

- The idea of a decidable theory
- Thm. 3: Any consistent, negation-complete, axiomatized formal theory is decidable
- Expressing and capturing properties
- The idea of a sufficiently strong theory
- Thm. 4: No consistent, sufficiently strong, axiomatized formal theory is decidable
- Why the argument for Thm. 4 involves 'diagonalization'
- Hence, from Thms 3 and 4: A consistent, sufficiently strong, axiomatized formal theory cannot be negation complete

---

In Episode 1, we introduced the very idea of a negation-incomplete effectively formalized theory T.

We noted that if we are aiming to construct a theory of basic arithmetic, we would ideally like the theory to be able to prove *all* the truths expressible in the language of basic arithmetic, and hence to be negation complete (at least as far as

statements of basic arithmetic are concerned). But Gödel's First Incompleteness Theorem says, very roughly, that that's impossible: a nice enough theory T will always be negation incomplete for basic arithmetic.

Now, as we remarked, the Theorem comes in two flavours, depending on whether we cash out the idea of being 'nice enough' in terms of (i) the semantic idea of T's being a *sound theory which uses enough of the language of arithmetic*, or (ii) the idea of T's being a *consistent theory which proves enough arithmetic*. And we noted that Gödel's own proofs, of either flavour, go via the idea of numerically coding up inside arithmetic syntactic facts about what can be proved in T, and then constructing an arithmetical sentence that – via the coding – in effect 'says' *I am not provable in* T.

We ended by noting that, at least at the level of arm-waving description of Episode 1, the Gödelian construction might look a bit worrying. After all, we all know that self-reference is dangerous – think Liar Paradox! So is Gödel's construction entirely legitimate?

As I hope will become clear as we go along, it certainly is. But first I think it might well go a little way towards calming anxieties that some illegitimate trick is being pulled, and it is certainly of intrinsic interest, if we give a somewhat different sort of proof of incompleteness, one that doesn't go via any explicitly self-referential construction. This proof will, however, introduce the idea of a **diagonalization argument**: and as we will later see that it is in fact diagonalization rather than self-reference which is really the key to Gödel's own proof.

So now read on ...

2.1 Negation completeness and decidability

Let's start with another definition (sections, definitions and theorems will be numbered consecutively through these notes, to make cross-reference easier):

Defn. 14. *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 $T \vdash \varphi$.*

It's then easy to show:

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

Proof For convenience, we'll assume T's proof-system is a Frege/Hilbert axiomatic logic, where proofs are just linear sequences of wffs (it should be obvious

how to generalize the argument to other kinds of proof systems, e.g. where proof arrays are arranged as trees of some kind).

Recall, we stipulated (in Defns 2, 3) that if T is a properly formalized theory, its formalized language L has a finite number of basic symbols. Now, we can evidently put those basic symbols in some kind of 'alphabetical order', and then start mechanically listing off all the possible strings of symbols in some kind of order – e.g. the one-symbol strings, followed by the finite number of two-symbol strings in 'dictionary' order, followed by the finite number of three-symbol strings in 'dictionary' order, followed by the four-symbol strings, etc., etc.

Now, as we go along, generating sequences of symbols, it will be a mechanical matter to decide whether a given string is in fact a sequence of wffs. And if it is, it will be a mechanical matter to decide whether the sequence of wffs is a T-proof, i.e. check whether each wff is either an axiom or follows from earlier wffs in the sequence by one of T's rules of inference. (That's all effectively decidable in a properly formalized theory, by Defns 2, 3). If the sequence is indeed a kosher, well-constructed, proof, then list its last wff φ, i.e. the theorem proved.

So, we can in this way, start mechanically generating a list of all the T-theorems (since any T-theorem is the last wff in a proof).

And that enables us to decide, of an arbitrary sentence φ of our consistent, negation-complete T, whether it is indeed a T-theorem. Just start dumbly listing all the T-theorems. Since T is negation complete, eventually either φ or $\neg\varphi$ turns up (and then you can stop!). If φ turns up, declare it to be a theorem. If $\neg\varphi$ turns up, then since T is consistent, we can declare that φ is *not* a theorem.

Hence, there *is* a dumbly mechanical 'wait and see' procedure for deciding whether φ is a T-theorem, a procedure which (given our assumptions about T) is guaranteed to deliver a verdict in a finite number of steps. □

We are, of course, relying here on a very, *very*, relaxed notion of effective decidability-in-principle, where we aren't working under any practical time constraints or constraints on available memory etc. (so note, 'effective' doesn't mean 'practically efficacious' or 'efficient'!). 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 the official generous sense (explained more in *IGT2*, §3.1).

2.2 Capturing numerical properties in a theory

Here's an equivalent way of rewriting part of the earlier Defn. 13:

Defn. 15. *A property P is* **expressed** *by the open wff $\varphi(\mathsf{x})$ with one free variable in a language L which contains the language of basic arithmetic iff, for every n,*

i. if n has the property P, then $\varphi(\overline{\mathsf{n}})$ is true,

ii. if n does not have the property P, then $\neg\varphi(\overline{\mathsf{n}})$ is true.

(Recall, $\overline{\mathsf{n}}$ indicates L's standard numeral for n. And we won't fuss about the obvious extension to many-place relations and functions.) Now we want a new companion definition:

Defn. 16. *The theory T* **captures** *the property P by the open wff $\varphi(\mathsf{x})$ iff, for any n,*

i. if n has the property P, then $T \vdash \varphi(\overline{\mathsf{n}})$,

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

So: what a theory can *express* depends on the richness of its language; what a theory can *capture* – mnemonic: case-by-case prove – depends on the richness of its axioms and rules of inferences. ('Represents' is more commonly used than 'captures', but I'll stick here to the jargon adopted in *IGT2*.)

Ideally, of course, we'll want any theory that aims to deal with arithmetic not just to express but to capture lots of arithmetical properties, i.e. to prove which particular numbers have or lack these properties.

But what sort of properties do we want to capture? Well, 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 (see Defn. 1 again). So we can, in principle, run the procedure to decide whether n has this property P. 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 a property P. Formalization aims at regimenting what we can in principle already do: it isn't supposed to hobble our efforts. So while we might have some passing interest in more limited theories, we might 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 naturally will want T to capture P in the sense of our definition.

The working 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:

Defn. 17. *A formal theory T including some arithmetic is* **sufficiently strong** *iff it captures all decidable numerical properties.*

(It would be equally natural, of course, to require the theory also capture all decidable relations and all computable functions – but for present purposes we don't need to worry about that!)

To repeat: it seems a reasonable and desirable condition on an ideal formal theory of the arithmetic of the natural numbers that it be sufficiently strong: in effect, when *we* can (or at least, given world enough and time, *could*) decide whether a number has a certain property, the *theory* can do it.

2.3 Sufficiently strong theories are undecidable

We now prove a lovely theorem, our first significant result (take it slowly, savour it!):

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

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

If T is sufficiently strong, it must have a supply of open wffs (for expressing numerical properties). And by Defn 2, it must in fact be decidable what strings of symbols are open T-wffs with the free variable 'x'. And we can use the dodge in the proof of Theorem 3 to start mechanically listing such wffs

$$\varphi_0(\mathsf{x}), \varphi_1(\mathsf{x}), \varphi_2(\mathsf{x}), \varphi_3(\mathsf{x}), \ldots.$$

For we can just churn out all the strings of symbols of T's language 'in alphabetical order', and mechanically select out the wffs with free variable 'x'.

Now we can introduce the following definition:

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

That's a perfectly coherent stipulation. Of course, property D isn't presented in the familiar way in which we ordinarily present properties of numbers: but our

definition in a quite determinate way tells us whether n has the property D or not, and that's all we will need.[1]

Now for the key observation: our supposition that T is a decidable theory entails that D is an effectively decidable property of numbers.

Why? Well, given any number n, it will be a mechanical matter to start listing off the open wffs until we get to the n-th one, $\varphi_n(\mathsf{x})$. 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 eventually occur somewhere in our list 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, just 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 for the particular case $n = d$:

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.

[1] And note too that the fact that our *mode of presentation* of the property D is peculiar, is not 'ordinarily arithmetical', leaves it quite open that there could be an alternative mode of presentation of the same property which *is* 'ordinarily arithmetical'. In fact, Gödel coding tricks will ensure that there will be.

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

So, if T is properly formalized, consistent and can prove enough arithmetic, then there is no way of mechanically determining what's a T-theorem and what isn't. We could, I suppose, call this result a *non-trivialization theorem*. We can't trivialize an interesting area of mathematics which contains enough arithmetic by regimenting it into an effectively axiomatized theory T, and then just pass T over to a computer to tell us what's a theorem and what isn't.

Diagonalization Let's highlight the key construction here. We take a sequence of wffs $\varphi_n(\mathsf{x})$ (for $n = 0, 1, 2, \ldots$) and then consider the (negations of) the wffs $\varphi_0(\overline{0})$, $\varphi_1(\overline{1})$, $\varphi_2(\overline{2})$, etc. This sort of thing is called a *diagonalizing*. Why?

Well just imagine the square array you get by writing $\varphi_0(\overline{0})$, $\varphi_0(\overline{1})$, $\varphi_0(\overline{2})$, etc. in the first row, $\varphi_1(\overline{0})$, $\varphi_1(\overline{1})$, $\varphi_1(\overline{2})$, etc. in the next row, $\varphi_2(\overline{0})$, $\varphi_2(\overline{1})$, $\varphi_2(\overline{2})$ etc. in the next row, and so on [go on, draw the diagram!]. *Then the wffs of the form $\varphi_n(\overline{n})$ lie down the diagonal.*

We'll be meeting some other instances of this kind of construction. And it is 'diagonalization' in this sense that is really at the heart of Gödel's incompleteness proof.

2.4 An incompleteness proof

So we have shown:

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

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

It immediately follows that

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

Wonderful! A seemingly remarkable theorem proved remarkably quickly.

Though note that – unlike Gödel's own result – Theorem 5 doesn't actually yield a specific undecidable sentence for a given theory T. And more importantly, *the interest of the theorem entirely depends on the still-informal notion of a sufficiently strong theory being in good order.*

Well, obviously, I wouldn't have written this episode if the notion of sufficient strength was intrinsically problematic. Still, we are left with a major project here: we need to give a decent sharp account of what makes for a decidable property in order to (i) clarify the notion of sufficient strength, while (ii) still making it plausible that we want sufficiently strong theories in this clarified sense. The trick can be done, and it turns out that a very surprisingly weak theory called Robinson Arithmetic is already sufficiently strong. However, supplying and defending the needed sharp account of the notion of effective decidability in order to pin down the notion of sufficient strength takes quite a lot of effort! And it arguably takes more effort – or at least, more new ideas in addition to those already sketched – compared with filling in the needed details for proving incompleteness by Gödel's original method as partially sketched in Episode 1. So over the next episodes, we are going to revert to exploring Gödel's route to the incompleteness theorems. However, our informally-based argument in this present episode is highly suggestive and well worth knowing about.

Further reading

From *IGT2*, read §§2.3, 3.1–3.3 and then Chs 4, 5, and 7 (which isn't quite as much as it sounds!).

In these notes, we've skipped over the material in *IGT2* §§3.4-3.5 and Ch. 6 which together give us another way of informally proving an incompleteness theorem. It is up to you whether you tackle that material now or keep it until much later (in fact the arguments might be found by some to be a bit tricky, and I wavered about whether I include them so early in book: I certainly don't want you to get fazed by them – so skip if they are too puzzling at this stage).

If you have never seen the grandfather of all diagonalization arguments, due to Georg Cantor, then see http://en.wikipedia.org/wiki/Cantor's_diagonal_argument (as well as *IGT2*, §2.5). This anonymous pdf also says more about other diagonal arguments: http://wiki.laptop.org/images/2/28/Cantors_Diagonal_Method.pdf.

Episode 3

Two weak arithmetics

- Baby Arithmetic is negation complete
- Robinson Arithmetic, Q
- A simple proof that Robinson Arithmetic is not complete
- Adding $\leq$ to Robinson Arithmetic
- Why Robinson Arithmetic is interesting

Our last big theorem – Theorem 5 – tells us that *if* a theory meets certain conditions of 'sufficient strength', then it must be negation incomplete. And we made some initial arm-waving remarks to the effect that it *seems* plausible that we should want theories which meet those conditions. We then announced in passing that there actually *is* a weak arithmetic called Robinson Arithmetic which meets the conditions (in which case, stronger arithmetics will also meet the conditions). But we didn't say anything about what such a weak theory really looks like. In fact, we haven't looked at *any* detailed theory of arithmetic yet! It is high time, then, that we stop operating at the extreme level of abstraction of Episodes 1 and 2, and start getting our hands dirty.

This episode, then, introduces a couple of weak arithmetics ('arithmetics', that is to say, in the sense of 'theories of arithmetic'). We first meet Baby Arithmetic (as a warm-up) and then the important Robinson Arithmetic. Frankly, this in itself will be pretty unexciting stuff – by all means skip fairly lightly over

some of the boring proof details! But you do need to get a flavour of how these two simple formal theories work, in preparation for the next episode where we tackle the canonical first-order arithmetic PA.

3.1 Baby Arithmetic

We start by looking at an evidently sound theory BA ('Baby Arithmetic'). *This is a* **negation complete** *theory of arithmetic.*

How is that possible? – for recall

Theorem 1. *If T is a sound formal axiomatized theory whose language contains the language of basic arithmetic, then there will be a true sentence G_T of basic arithmetic such that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$, so T is negation incomplete.*

Answer: BA's very limited language L_B lacks quantifiers, so doesn't fully contain the language of basic arithmetic (see again Defn. 9, where we said that containing the language of basic arithmetic involved having the usual first-order logical apparatus). Because its language is so limited, Baby Arithmetic can in fact prove or disprove every sentence constructible in its restricted language L_B.

3.1.1 The language L_B

The language L_B has non-logical symbols

$$\mathsf{0}, \mathsf{S}, +, \times$$

The first of these is of course a constant (intended to denoted zero). The next symbol is a one-place function symbol (intended to denote the successor function). The last two symbols in the list are two-place function symbols (with the obvious standard interpretations). Note that if we use '+' and '×' as 'infix' function symbols in the usual way – i.e. we write $\mathsf{S0} + \mathsf{SS0}$ rather than prefix the function sign as in $+\,\mathsf{S0}\,\mathsf{SS0}$ – then we'll also need brackets for scoping the function signs, to disambiguate $\mathsf{S0} + \mathsf{SS0} \times \mathsf{SSS0}$, e.g. as $(\mathsf{S0} + (\mathsf{SS0} \times \mathsf{SSS0}))$.

From these symbols, we can construct the *terms* of L_B. A term is a referring expression built up from occurrences of '0' and applications of the function expressions 'S', '+', '×'. So, examples are $\mathsf{0}$, $\mathsf{SSS0}$, $(\mathsf{S0} + \mathsf{SS0})$, $((\mathsf{S0} + \mathsf{SS0}) \times \mathsf{SSS0})$.

The *value* of a term is the number it denotes when standardly interpreted: the values of our example terms are respectively 0, 3, 3 and 9.

Recall, we use '$\overline{\mathsf{n}}$' to represent the standard numeral $\mathsf{SS}\ldots\mathsf{S0}$ with n occurrences of 'S'. Thus '$\overline{\mathsf{3}}$' is short for '$\mathsf{SSS0}$'. The value of '$\overline{\mathsf{n}}$' is of course n.

The sole built-in predicate of the language L_B is the logical identity sign. Since L_B lacks other non-logical predicates, its only way of forming atomic wffs is therefore by taking two terms constructed from the non-logical symbols and putting the identity sign between them. In other words, the atomic wffs of L_B are *equations* relating terms denoting particular numbers. So, for example, $(\mathsf{S0} + \mathsf{SS0}) = \mathsf{SSS0}$ is a true atomic wff – which we can abbreviate, dropping brackets in a natural way, as $\overline{1} + \overline{2} = \overline{3}$ – and $(\mathsf{S0} \times \mathsf{SS0}) = \mathsf{SSS0}$ is a false one.

We also, however, want to be able to express *inequations*, hence we'll want L_B to have a negation sign. And note, for convenience, we will abbreviate wffs of the form $\neg\, \tau_1 = \tau_2$ by $\tau_1 \neq \tau_2$.

In *IGT2*, I go on to round things out so as to give L_B some expressively complete set of propositional connectives, e.g. $\neg, \wedge, \vee, \rightarrow$. We'll then also of course need brackets again for scoping the two-place connectives if we give them an 'infix' syntax in the familiar way.

The syntax for constructing the complete class of wffs of L_B is then exactly as you'd expect, and the semantics is the obvious one. [Exercise: spell out the details carefully!]

3.1.2 The axioms and logic of BA

The theory BA in the language L_B comes equipped with some classical propositional deductive system to deal with the propositional connectives (choose your favourite system!) and the usual identity rules.

Next, we want non-logical axioms governing the successor function. We want to capture the ideas that, if we start from zero and repeatedly apply the successor function, we keep on getting further numbers – i.e. different numbers have different successors: contraposing, for any m, n, if $Sm = Sn$ then $m = n$. And further, we never cycle back to zero: for any n, $0 \neq Sn$.

However, there are no quantifiers in L_B. So we can't directly express those general facts about the successor function inside the object language L_B. Rather, we have to employ **schemata** (i.e. general templates) and use the generalizing apparatus in our English metalanguage to say: *any sentence that you get from one of the following schemata by substituting standard numerals for the placeholders 'ζ', 'ξ' is an axiom.*

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

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

Next, we want non-logical axioms for addition. This time we want to capture the idea that adding zero to a number makes no difference: for any m, $m+0 = m$. And adding a non-zero number Sn to m is governed by the rule: for any m, n, $m + Sn = S(m + n)$. Those two principles together tell us how to add zero to a given number m; and then adding one is defined as the successor of the result of adding zero; and then adding two is defined as the successor of the result of adding one; and so on up – thus defining adding n for any particular natural number n.

Note, however, because of its lack of quantifiers, we can't express all that directly inside L_B. We have to resort to schemata again, and say that *anything you get by substituting standard numerals for placeholders in the following is an axiom:*

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

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

We can similarly pin down the multiplication function by requiring that *every numeral instance of the following is an axiom too*:

Schema 5. $\zeta \times \mathsf{0} = \mathsf{0}$

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

Instances of Schema 5 tell us the result of multiplying by zero. Instances of Schema 6 with 'ξ' replaced by '$\mathsf{0}$' 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 '$\mathsf{S0}$' 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.

To summarize, then,

Defn. 18. BA *is the theory whose language is L_B, whose logic is classical propositional logic plus standard identity rules, and whose non-logical axioms are every numerical instance of Schemas (1) to (6).*

Notation again Recall our convention from the box on p. 12: sans serif expressions belong to L_B, or whatever formal language is under discussion: *italic* symbols are just part of everyday mathematical English.

And the likes of 'ξ' are metalinguistic variable, here for numerals of L_B, later for terms of the language L_A more generally (including object-language variables). Such Greek letters are *not* part of the language L_B (or L_A). Rather they are added to informal mathematical English to expedite our talking about wffs in the formal languages. And, if you are really pernickety, you could/should write, for example,

for any numerals ζ, ξ, $\ulcorner \mathsf{S}\zeta = \mathsf{S}\xi \to \zeta = \xi \urcorner$ is a BA-axiom,

where the corners are Quine quotes (see http://en.wikipedia.org/wiki/Quasi-quotation).

Ordinary mathematical English is of course casual about such quotational niceties, and often recruits object-language variables x, y etc. to do the work of expressing schematic generalizations. That's usually entirely harmless, but in these notes we'll not-very-consistently try sometimes to be a bit more careful – hence the use of Greek letters for metalinguistic variables – even if we don't get ultra careful and go the whole hog with Quine quotes (and allow ourselves to write, e.g., $\varphi(\overline{\mathsf{n}})$ without fancy quotes either).

3.2 Some proofs inside BA, and three little theorems

We'll give two little examples of how arithmetic can be done inside BA. First, let's show that $\mathsf{BA} \vdash \overline{4} \neq \overline{2}$, i.e. $\mathsf{BA} \vdash \mathsf{SSSS0} \neq \mathsf{SS0}$

1.	$\mathsf{SSSS0} = \mathsf{SS0}$	Supposition
2.	$\mathsf{SSSS0} = \mathsf{SS0} \to \mathsf{SSS0} = \mathsf{S0}$	Axiom, instance of Scher
3.	$\mathsf{SSS0} = \mathsf{S0}$	From 1, 2 by MP
4.	$\mathsf{SSS0} = \mathsf{S0} \to \mathsf{SS0} = \mathsf{0}$	Axiom, instance of Scher
5.	$\mathsf{SS0} = \mathsf{0}$	From 3, 4 by MP
6.	$\mathsf{0} \neq \mathsf{SS0}$	Axiom, instance of Scher
7.	Contradiction!	From 5, 6 and identity ru
8.	$\mathsf{SSSS0} \neq \mathsf{SS0}$	From 1 to 7, by RAA.

And a little reflection on that illustrative proof should now convince you of this general claim:

Theorem 6. *If s and t are distinct numbers, then* $\mathsf{BA} \vdash \overline{\mathsf{s}} \neq \overline{\mathsf{t}}$.

[Exercise for the mathematical: turn that 'little reflection' into a proper proof!]

And for our second example, we'll show that $\mathsf{BA} \vdash \overline{2} \times \overline{1} = \overline{2}$. In unabbreviated form, though dropping outermost brackets, we need to derive $\mathsf{SS0} \times \mathsf{S0} = \mathsf{SS0}$.

1.	$\mathsf{SS0} \times 0 = 0$	Axiom, instance of Scher
2.	$\mathsf{SS0} \times \mathsf{S0} = (\mathsf{SS0} \times 0) + \mathsf{SS0}$	Axiom, instance of Scher
3.	$\mathsf{SS0} \times \mathsf{S0} = 0 + \mathsf{SS0}$	From 1, 2 by LL

('LL' of course indicates the use of Leibniz's Law 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 we have to do a bit of work to get it:

4.	$0 + 0 = 0$	Axiom, instance of Scher
5.	$0 + \mathsf{S0} = \mathsf{S}(0 + 0)$	Axiom, instance of Scher
6.	$0 + \mathsf{S0} = \mathsf{S0}$	From 4, 5 by LL
7.	$0 + \mathsf{SS0} = \mathsf{S}(0 + \mathsf{S0})$	Axiom, instance of Scher
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, to be sure, but again it works. And inspection of BA's axioms and a little reflection on our second illustrative proof should now convince you of a further general claim:

Theorem 7. BA *can prove any true equation of the form* $\overline{\mathsf{m}} + \overline{\mathsf{n}} = \overline{\mathsf{t}}$ *or* $\overline{\mathsf{m}} \times \overline{\mathsf{n}} = \overline{\mathsf{t}}$.

In other words, BA can correctly add or multiply any two numbers. [Exercise: give a proper proof of that!]

We can now generalize further: in fact BA can correctly evaluate all terms of its language. That is to say,

Theorem 8. *Suppose τ is a term of L_B and the value of τ on the intended interpretation of the symbols is t. Then* $\mathsf{BA} \vdash \tau = \overline{\mathsf{t}}$.

Why so? Well, let's take a very simple example and then draw a general moral. Suppose we want to show e.g. that $(\overline{2} + \overline{3}) \times (\overline{2} \times \overline{2}) = \overline{20}$. Then evidently we'll proceed as follows.

1.	$(\overline{2}+\overline{3})\times(\overline{2}\times\overline{2})=(\overline{2}+\overline{3})\times(\overline{2}\times\overline{2})$	Trivial!
2.	$\overline{2}+\overline{3}=\overline{5}$	Instance of Thm. 7
3.	$(\overline{2}+\overline{3})\times(\overline{2}\times\overline{2})=\overline{5}\times(\overline{2}\times\overline{2})$	From 1, 2 using LL
4.	$\overline{2}\times\overline{2}=\overline{4}$	Instance of Thm. 7
5.	$(\overline{2}+\overline{3})\times(\overline{2}\times\overline{2})=\overline{5}\times\overline{4}$	From 3, 4 using LL
6.	$\overline{5}\times\overline{4}=\overline{20}$	Instance of Thm. 7
7.	$(\overline{2}+\overline{3})\times(\overline{2}\times\overline{2})=\overline{20}$	From 5, 6 using LL

What we do here is evidently 'evaluate' the complex formula on the left 'from the inside out', reducing the complexity of what needs to evaluated at each stage, eventually equating the complex formula with a standard numeral. [Exercise, for those who like this sort of thing: give a proper argument 'by induction on the complexity of the formula' that proves Thm 8.]

3.3 BA is a sound and complete theory of the truths of L_B

Our little theorems now enable us to prove the following:

Theorem 9. *Suppose σ and τ are terms of L_B. Then if $\sigma = \tau$ is true, then $\mathsf{BA} \vdash \sigma = \tau$, and if $\sigma = \tau$ is false, then $\mathsf{BA} \vdash \sigma \neq \tau$.*

Proof Let σ evaluate to s and τ evaluate to t. Then, by Theorem 8, (i) $\mathsf{BA} \vdash \sigma = \overline{\mathsf{s}}$ and (ii) $\mathsf{BA} \vdash \tau = \overline{\mathsf{t}}$.

Now, suppose $\sigma = \tau$ is true. Then $s = t$, and so $\overline{\mathsf{s}}$ must be the very same numeral as $\overline{\mathsf{t}}$. We can therefore immediately conclude from (i) and (ii) that $\mathsf{BA} \vdash \sigma = \tau$ by the logic of identity.

Suppose, on the other hand, that $\sigma = \tau$ is false, so $s \neq t$. Then by Theorem 6, $\mathsf{BA} \vdash \overline{\mathsf{s}} \neq \overline{\mathsf{t}}$, and together with (i) and (ii) that implies $\mathsf{BA} \vdash \sigma \neq \tau$, again by the logic of identity. □

And from that last theorem, it more or less immediately follows that

Theorem 10. BA *is negation complete.*

Proof The only atomic claims expressible in BA are equations involving 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 true negation of each false equation.

But now recall that there's a theorem of propositional logic which tells us that, given some atoms and/or negated atoms, we can prove every complex wff that must be true if those atoms/negated atoms are true, and prove the negation of every complex wff that must be false if those atoms/negated atoms are true. That means, given (1) and (2), we can derive any true truth-functional combination of the equations/inequations in a complex wff, i.e. prove any true sentence. Likewise, we can also derive the negation of any false truth-functional combination of the equations/inequations in a complex wff, i.e. prove the negation of any false sentence.

Hence, for *any* sentence φ of BA, since either φ is true or false, either $\mathsf{BA} \vdash \varphi$ or $\mathsf{BA} \vdash \neg\varphi$. Hence BA is negation complete. □

So the situation is this. BA is obviously a sound theory – all its axioms are trivial arithmetical truths, and its logic is truth-preserving, so all its theorems are true. BA is also, as we've just seen, a complete theory in the sense of entailing all the truths expressible in its language L_B. However, the language L_B only allows us to express a limited class of facts about adding and multiplying particular numbers (it can't express numerical generalizations). And, prescinding from practical issues about memory or display size, any pocket calculator can in effect tell us about all such facts. So it is no surprise that we can get a formalized theory to do the same!

3.4 Robinson Arithmetic, Q

That's all very straightforward, but also very unexciting. 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, as we stressed, 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 interesting is to restore the familiar apparatus of quantifiers and variables. So that's what we'll start doing.

3.4.1 The language L_A

We'll keep the same non-logical vocabulary as in L_B: so there is still just a single non-logical constant denoting zero, and the three built-in function-symbols, $\mathsf{S}, +, \times$ expressing successor, addition and multiplication. But now we allow our-

selves the full linguistic resources of first-order logic, with the usual supply of quantifiers and variables to express generality. We fix the domain of the quantifiers to be the natural numbers. The result is the interpreted language L_A.

L_A is the least ambitious language which 'contains the language of basic arithmetic' in the sense of Defn. 9. (For, of course, L_A has the the predicate expression '$\mathsf{x} = \mathsf{x}$' which has the numbers as its extension, so fits our official definition, if we want to fuss about that.)

3.4.2 The axioms and logic of Q

The theory Q, **Robinson Arithmetic**, is built in the formal language L_A, and is equipped with a full first-order classical logic. And as for the non-logical axioms, now we have the quantifiers available to express generality we can replace each of BA's metalinguistic schemata (specifying an infinite number of formal axioms governing particular numbers) by a single generalized Axiom expressed inside L_A itself. For example, we can replace the first two schemata governing the successor function by

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

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

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

These Axioms tell us that zero isn't a successor, but they don't explicitly rule it out that there are other objects that aren't successors cluttering up the domain of quantification (i.e. there could be 'pseudo-zeros'). We didn't need to fuss about this before, because by construction BA can only talk about the numbers represented by standard numerals in the sequence '$\mathsf{0}, \mathsf{S0}, \mathsf{SS0}, \ldots$'. But now we have the quantifiers in play. And these quantifiers are intended to run over the natural numbers – we certainly don't intend them to be running over stray objects other than zero that aren't successors. So let's reflect that in our axioms by explicitly ruling out such strays:

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

Next, we can similarly replace our previous schemata for addition and multiplication by universally quantified Axioms in the obvious way:

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

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

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

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

Again, each of these axioms entails all the instances of BA's corresponding schema.

Defn. 19. *The formal axiomatized theory with language L_A, Axioms 1 to 7, plus a classical first-order logic, is standardly called* **Robinson Arithmetic***, or simply* Q*.*

It is worth noting, for future reference, that it was first isolated as a weak system of arithmetic worthy of study by Raphael M. Robinson in 1952 – i.e. long after Gödelian incompleteness was discovered.

3.5 Robinson Arithmetic is not complete

Q is assuredly 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 of L_A are theorems?

Since any old BA axiom – i.e. any instance of one of our previous schemata – can be derived from one of our new Q Axioms, every L_B-sentence that can be proved in BA is equally a quantifier-free L_A-sentence which can be proved in Q. Hence,

Theorem 11. Q *correctly decides every quantifier-free L_A sentence (i.e.* $Q \vdash \varphi$ *if the quantifier-free wff φ is true, and* $Q \vdash \neg\varphi$ *if the quantifier-free wff φ is false).*

So far, so good. However, there are very simple true quantified sentences that Q can't prove. For example, Q can of course prove any particular wff of the form $0 + \overline{n} = \overline{n}$. *But it can't prove the corresponding universal generalization*:

Theorem 12. $Q \nvdash \forall x(0 + x = x)$.

Proof Since Q is a theory with a standard first-order theory, for any L_A-sentence φ, $Q \vdash \varphi$ if and only if $Q \vDash \varphi$ (that's just the completeness theorem for first-order logic). Hence one way of showing that $Q \nvdash \varphi$ is to show that $Q \nvDash \varphi$: and we can show *that* by producing a countermodel to the entailment – i.e. by finding an interpretation (a deviant, unintended, 'non-standard', re-interpretation) for

L_A's wffs which makes Q's axioms true-on-that-interpretation but which makes φ false.

So here goes: take the domain of our deviant, unintended, re-interpretation 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 – but any other pair of distinct non-numbers will do). 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 very easy to check that Axioms 1 to 3 are still true on this deviant interpretation. Zero is still not a successor. Different elements have different successors. And every non-zero element is a successor (perhaps a self-successor!).

We now need to extend this interpretation to re-interpret the function-symbol '+'. 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 $x +^* b = a$. If you prefer that in a matrix (read off *row* $+^*$ *column*):

$+^*$	n	a	b
m	$m+n$	b	a
a	a	b	a
b	b	b	a

It is again easily checked that interpreting '+' as addition* still makes Axioms 4 and 5 true. (In headline terms: For Axiom 4, we note that adding* zero on the right always has no effect. For Axiom 5, just consider cases. (i) $m +^* S^*n = m + Sn = S(m + n) = S^*(m +^* n)$ for 'ordinary' numbers m, n in the domain. (ii) $a + S^*n = a = S^*a = S^*(a +^* n)$, for 'ordinary' n. Likewise, (iii) $b + S^*n = S^*(b +^* n)$. (iv) $x +^* S^*a = x + a = b = S^*b = S^*(x +^* a)$, for any x in the domain. (v) Finally, $x +^* S^*b = S^*(x +^* b)$. Which covers every possibility.)

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 $\forall \mathsf{x}(\mathsf{0} + \mathsf{x} = \mathsf{x})$ false. So $\mathsf{Q} \nvdash \forall \mathsf{x}(\mathsf{0} + \mathsf{x} = \mathsf{x})$ □

Theorem 13. Q *is negation-incomplete.*

Proof. Put $\varphi = \forall \mathsf{x}(\mathsf{0} + \mathsf{x} = \mathsf{x})$. We've just shown that $\mathsf{Q} \nvdash \varphi$. But obviously, Q can't prove $\neg\varphi$ either. Just revert to the standard interpretation built into L_A. Q

certainly has true axioms on this interpretation. So all theorems are true on that interpretation, but $\neg\varphi$ is false on that interpretation, so it can't be a theorem. Hence φ is formally undecidable in Q. □

Of course, we've already announced that Gödel's incompleteness theorem is going to prove that *no* sound axiomatized theory whose language is at least as rich as L_A can be negation complete – that was Theorem 1. But 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.

3.6 Statements of order in Robinson Arithmetic

We now see how to define the usual linear ordering on the numbers in Robinson Arithmetic by an expression that means **less than or equals to**. In fact, we can do better than 'expressing' this order relation: for

Theorem 14. *In* Q*, the* less-than-or-equal-to *relation is captured by the wff* $\exists\mathsf{v}(\mathsf{v}+\mathsf{x}=\mathsf{y})$.

We need to show that, for any particular pair of numbers, m, n, if $m \leq n$, then $\mathsf{Q} \vdash \exists\mathsf{v}(\mathsf{v}+\overline{\mathsf{m}}=\overline{\mathsf{n}})$, and otherwise $\mathsf{Q} \vdash \neg\exists\mathsf{v}(\mathsf{v}+\overline{\mathsf{m}}=\overline{\mathsf{n}})$.

Proof 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 addition sum. So we have $\mathsf{Q} \vdash \overline{\mathsf{k}}+\overline{\mathsf{m}}=\overline{\mathsf{n}}$. But logic gives us $\overline{\mathsf{k}}+\overline{\mathsf{m}}=\overline{\mathsf{n}} \vdash \exists\mathsf{v}(\mathsf{v}+\overline{\mathsf{m}}=\overline{\mathsf{n}})$ by existential quantifier introduction. Therefore $\mathsf{Q} \vdash \exists\mathsf{v}(\mathsf{v}+\overline{\mathsf{m}}=\overline{\mathsf{n}})$, as was to be shown.

Suppose alternatively $m > n$. We need to show $\mathsf{Q} \vdash \neg\exists\mathsf{v}(\mathsf{v}+\overline{\mathsf{m}}=\overline{\mathsf{n}})$. We'll first demonstrate this in the case where $m=2$, $n=1$, using a Fitch-style proof-system. For brevity we will omit statements of Q's axioms and some other trivial steps; we drop unnecessary brackets

1.	$\exists\mathsf{v}(\mathsf{v}+\mathsf{SS0}=\mathsf{S0})$	Supposition
2.	$\quad\mathsf{a}+\mathsf{SS0}=\mathsf{S0}$	Supposition
3.	$\quad\mathsf{a}+\mathsf{SS0}=\mathsf{S}(\mathsf{a}+\mathsf{S0})$	From Axiom 5
4.	$\quad\mathsf{S}(\mathsf{a}+\mathsf{S0})=\mathsf{S0}$	From 2, 3 by LL
5.	$\quad\mathsf{a}+\mathsf{S0}=\mathsf{S}(\mathsf{a}+0)$	From Axiom 5
6.	$\quad\mathsf{SS}(\mathsf{a}+0)=\mathsf{S0}$	From 4, 5 by LL
7.	$\quad\mathsf{a}+0=\mathsf{a}$	From Axiom 4
8.	$\quad\mathsf{SSa}=\mathsf{S0}$	From 6, 7 by LL
9.	$\quad\mathsf{SSa}=\mathsf{S0}\rightarrow\mathsf{Sa}=0$	From Axiom 2

10.	$\mathsf{Sa} = \mathsf{0}$	From 8, 9 by MP
11.	$\mathsf{0} = \mathsf{Sa}$	From 10
12.	$\mathsf{0} \neq \mathsf{Sa}$	From Axiom 1
13.	Contradiction!	From 11, 12
14.	Contradiction!	$\exists$E 1, 2–13
15.	$\neg\exists\mathsf{v}(\mathsf{v} + \mathsf{SS0} = \mathsf{S0})$	RAA 1–14

The only step to explain is at line (14) where we use a version of the Existential Elimination rule: if the temporary supposition $\varphi(\mathsf{a})$ leads to contradiction, for arbitrary a, then $\exists\mathsf{v}\varphi(\mathsf{v})$ 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 $\mathsf{Q} \vdash \neg\exists\mathsf{v}(\mathsf{v} + \overline{\mathsf{m}} = \overline{\mathsf{n}})$ whenever $m > n$. [Exercise: convince yourself that this claim is true!] So we are done. □

Given the result we've just proved, we can sensibly add the standard symbol '$\leq$' to L_A, the language of Q, defined so that whatever terms [NB, not just numerals] we put for 'ξ' and 'ζ', $\xi \leq \zeta$ is just short for $\exists\mathsf{v}(\mathsf{v} + \xi = \zeta)$, and then Q will be able to prove at least the expected facts about the less-than-or-equals relations among quantifier-free terms. (Well, 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.)

Note, by the way, that some presentations in fact treat '$\leq$' as a primitive symbol built into our formal theories like Q from the start, governed by its own additional axiom(s). But nothing important hangs on the difference between that approach and our policy of introducing the symbol by definition. (And of course, nothing hangs either on our policy of introducing '$\leq$' as our basic symbol rather than '$<$', which could have been defined by $\xi < \zeta =_{\text{def}} \exists\mathsf{v}(\mathsf{Sv} + \xi = \zeta)$.)

Since it so greatly helps readability, we'll henceforth make very free use of '$\leq$' as an 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 '$\leq$' is to be unpacked as we've just explained. And it is standard to further abbreviate such wffs by $(\forall\xi \leq \kappa)\varphi(\xi)$ and $(\exists\xi \leq \kappa)\varphi(\xi)$ respectively.

3.7 Why Q is interesting

Given it can't even prove $\forall x(0 + x = x)$, Q is evidently a *very* weak theory of arithmetic. Which is probably no surprise as (apart from Axiom 3) we've added little *axiomatic* proof-power to BA while adding a lot of *expressive power* to its language by adding quantifiers. So it's only to be expected that there will be lots of newly expressible truths that Q can't prove (and since Q is sound, it won't be able to disprove these truths either).

Even so, despite its great shortcomings, Q does have some nice properties. As we saw, it can capture the decidable relation that obtains when one number is at least as big as another. Moreover, we can eventually show that rather stunningly general result

Theorem 15. Q *can capture* all *decidable numerical properties – i.e. it is sufficiently strong in the sense of Defn 17.*

That might initially seem very surprising indeed, given Q's weakness. But remember, 'sufficient strength' was defined as a matter of being able to *case-by-case* prove enough wffs about decidable properties of individual numbers. It turns out that Q's hopeless weakness at proving generalizations doesn't stop it doing that.

So that's why Q is particularly interesting – it is about the weakest arithmetic which is sufficiently strong (and it was isolated by Robinson for that reason), and for which Gödelian proofs of incompleteness can be run. Suppose, then, that a theory is formally axiomatized, consistent and can prove everything Q can prove (those do indeed seem very modest requirements). Then what we've just announced and promised can be proved is that any such theory will be 'sufficiently strong'. And therefore e.g. Theorem 5 will apply – any such theory will be incomplete.

However, we can only establish that Q *does* have sufficient strength to capture all decidable properties if and when we have a quite general theory of decidability to hand. And we don't want to get embroiled in that (at least yet). So what we *will* be proving quite soon (in Episode 6) is a rather weaker claim about Q. We'll show that it can capture all so-called 'primitive recursive' properties, where these form a very 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: it means that Q gives us 'the modest amount of arithmetic' need for Theorem 2.

But before we get round to showing all this, we are first going to take a look at a *much* richer arithmetic than Q, namely PA.

Further reading

For parallel reading to this episode, see *IGT2*, Ch. 10, and also §11.1.

Visit http://en.wikipedia.org/wiki/Robinson_arithmetic to learn a little more about Q. For a book chapter at about the same level as *IGT2*, see Richard L. Epstein and Walter Carnielli, *Computability: Computable Functions, Logic, and the Foundatoins of Mathematics* (Wadsworth, 2000; Advanced Reasoning Forum, 2008), Ch. 21.

Episode 4

First-order Peano Arithmetic

- The ω-rule
- Induction: the induction axiom, the induction rule, the induction schema
- First-order Peano Arithmetic
- Why we might have expected PA to be negation complete

Here's the story so far. We noted in Episode 1 that Gödel showed, more or less,

Theorem 1. *If T is a sound formal axiomatized theory whose language contains the language of basic arithmetic, then there will be a true sentence G_T of basic arithmetic such that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$, so T is negation incomplete.*

Of course, we didn't *prove* that theorem, though we waved an arm airily at the basic trick that Gödel uses to establish the theorem – namely we 'arithmetize syntax' (i.e. numerically code up facts about provability in formal theorems) and then construct a Gödel sentence that sort-of-says 'I am not provable'.

We did note, however, that this theorem invokes the assumption that we dealing with a sound theory, and of course soundness is a *semantic* notion. For various reasons, Gödel thought it essential to establish that we can get incompleteness making merely *syntactic* assumptions, thus:

Theorem 2. *For any consistent formal axiomatized theory T which can prove a certain modest amount of arithmetic (and has a certain additional desirable syntactically definable property that any sensible formalized arithmetic will share), there is a sentence of basic arithmetic G_T such that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$, so T is negation incomplete.*

This theorem with syntactic assumptions is the sort of thing that's usually referred to as The First Incompleteness Theorem, and of course we again didn't prove it. Indeed, we didn't even say what that 'modest amount of arithmetic' is (nor did we say anything about that 'additional desirable property').

So Episode 1 was little more than a gesture in the right direction. In Episode 2, we did a bit better, in the sense that we actually gave a *proof* of the following theorem:

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

The argument was nice, as it shows that we can get incompleteness results without calling on the arithmetization of syntax and the construction of Gödel sentences. However the argument depended on working with the notion of 'sufficient strength' which is defined in terms of the an informal, intuitive, notion of a 'decidable property'. That's not in itself wicked, because lots of mathematical proofs involve informal, intuitive, concepts. But the discussion in Episode 2 doesn't give us any clue about how we can sharpen up the intuitive notions in play or about what a 'sufficiently strong' theory might look like.

Episode 3 took a step towards telling us what makes for a sufficiently strong (and likewise towards telling us what the 'modest amount of arithmetic' mentioned in Theorem 2 amounts to).

As a warm-up exercise, we first looked at BA, the quantifier-free arithmetic of the addition and multiplication of particular numbers. This is a complete (and hence decidable!) theory – but of course it is only complete, i.e. able to decide every sentence constructible in its language, because its language is indeed so weak.

If we augment the language of BA by allowing ourselves the usual apparatus of first-order quantification, and replace the schematically presented axioms of BA with their obvious universally quantified correlates (and add in the axiom that every number bar zero is a successor) we get Robinson Arithmetic Q. Since we've added pretty minimally to what is given in the axioms of BA while considerably enriching its language, it is probably no surprise that we have

Theorem 13. Q *is negation-incomplete.*

And we can prove this without any fancy Gödelian considerations. A familiar and simple kind of model-theoretic argument is enough to do the trick: we find a deviant interpretation of Q's syntax which is such as to make the axioms all true but on which $\forall x(0 + x = x)$ is false, thus establishing $Q \nvdash \forall x(0 + x = x)$. And since Q is sound on the built-in interpretation of its language, we also have $Q \nvdash \neg\forall x(0 + x = x)$.

Q, then, is a very weak arithmetic. Still, it will turn out to be the 'modest amount of arithmetic' needed to get Theorem 2 to fly. And we announced (we didn't prove it!)

Theorem 15. Q *can capture* all *decidable numerical properties – i.e. it is sufficiently strong in the sense of Defn 17.*

So a theory's containing Q makes it a 'sufficiently strong' theory in the sense of Theorem 5. Of course *establishing* this last Theorem is a non-trivial task for later: but we have at least explained why Q turns out to be so interesting despite its weakness.

Now read on ....

4.1 Arithmetical Induction

In this episode, we introduce the canonical first-order theory of arithmetic, PA. It's what you get by adding an **induction principle** to Q. So first, in this section, we need to say something about induction.

Let's go back to our proof that Q is negation-incomplete. Put $\psi(x)$ for $(0 + x = x)$. Then, as we noted, for any particular n, $Q \vdash \psi(\overline{n})$, for Q can prove an unquantified true equation. But we showed that $Q \nvdash \forall x\psi(x)$. In other words, Q can separately prove all instances of $\psi(\overline{n})$ but can't put everything together and prove the corresponding simple generalization. Let's now consider what proof-principle we might add to Q to fill this sort of yawning gap.

4.1.1 The ω-rule

Q, to repeat, proves each of $\psi(\overline{0}), \psi(\overline{1}), \psi(\overline{2}), \psi(\overline{3}), \ldots$. Suppose then that we added to Q the rule that we can infer as follows:

$$\frac{\begin{array}{ccccc}\vdots & \vdots & \vdots & \vdots & \\ \psi(\overline{0}) & \psi(\overline{1}) & \psi(\overline{2}) & \psi(\overline{3}) & \vdots\end{array}}{\forall \mathsf{x}\psi(\mathsf{x})}$$

This rule – or rather (of course) the generalized version, which says this inference pattern is valid for any ψ – is what's called **the ω-rule**. It is evidently a sound rule: if each $\psi(\overline{\mathsf{n}})$ is true then indeed all numbers n satisfy $\psi(\mathsf{x})$. Adding the ω-rule would certainly repair the gap we exposed in Q.

But of course there's a very big snag: the ω-rule is **infinitary**. It takes as input an infinite number of premisses. So proofs invoking this ω-rule will be infinite arrays. And being infinite, they cannot be mechanically checked in a finite number of steps to be constructed according to our expanded rules. In sum, then, a theory with a proof-system that includes an infinitary ω-rule can't count as an effectively axiomatized formal theory according to Defn. 3.

There is certainly some technical interest in investigating infinitary logics which allow infinitely long sentences (e.g. infinite conjunctions) and/or infinite-array proofs. But there is a clear sense in which such logics are not of practical use, and cannot be used to regiment how we finite beings in fact argue. The requirement of effective checkability of proofs which we imposed on formalized theories is, for that reason, not arbitrary. And so we'll stick to that requirement, and have to ban appeal to the ω-rule, at least in official proofs.

4.1.2 Replacing an infinitary rule with a finite one

To repeat, as well as proving $\psi(0)$, Q also proves $\psi(\overline{1}), \psi(\overline{2}), \psi(\overline{3}), \ldots$. And now note that it isn't, so to speak, a global accident that Q can prove all those. Rather, Q proves them in a uniform way.

One way to bring this out is to note first that we have the following proof in Q:

1.	$\psi(\mathsf{a})$	Supposition
2.	$0 + \mathsf{a} = \mathsf{a}$	Unpacking the definition
3.	$\mathsf{S}(0 + \mathsf{a}) = \mathsf{Sa}$	From 2 by the identity laws
4.	$(0 + \mathsf{Sa}) = \mathsf{S}(0 + \mathsf{a})$	Instance of Axiom 5
5.	$(0 + \mathsf{Sa}) = \mathsf{Sa}$	From 3, 4
6.	$\psi(\mathsf{Sa})$	Applying the definition
7.	$(\psi(\mathsf{a}) \rightarrow \psi(\mathsf{Sa}))$	From 1, 6 by Conditional Proof
8.	$\forall \mathsf{x}(\psi(\mathsf{x}) \rightarrow \psi(\mathsf{Sx}))$	From 7, since a was arbitrary.

Now, since Q trivially proves $\psi(0)$, we can appeal to our result $\forall x(\psi(x) \to \psi(Sx))$ to derive $\psi(0) \to \psi(\overline{1})$, and so modus ponens gives us $\psi(\overline{1})$. The same generalization also gives us $\psi(\overline{1}) \to \psi(\overline{2})$, so another modus ponens gives us $\psi(\overline{2})$. Now we can appeal to our generalization again to get $\psi(\overline{2}) \to \psi(\overline{3})$, and so can derive $\psi(\overline{3})$. Keep on going!

In this way, $\psi(0)$ and $\forall x(\psi(x) \to \psi(Sx))$ together prove each of of $\psi(\overline{0}), \psi(\overline{1}), \psi(\overline{2}), \ldots$, and those truths in turn, by the sound ω-rule, entail $\forall x\psi(x)$. Thus putting everything together we have the infinitary proof...

$$\dfrac{\dfrac{\psi(0) \quad \dfrac{\forall x(\psi(x) \to \psi(Sx)) \quad \psi(0)}{(\psi(0) \to \psi(S0))}}{\psi(S0)} \quad \dfrac{\dfrac{\forall x(\psi(x) \to \psi(Sx))}{(\psi(0) \to \psi(S0))} \quad \dfrac{\forall x(\psi(x) \to \psi(Sx))}{(\psi(S0) \to \psi(SS0))}}{\psi(SS0)}}{\psi(0)} \quad \ldots\ldots \quad \dfrac{}{\forall x\psi(x)}\ \omega\text{-}rule$$

But now suppose we cut out all those (infinitely many) intermediate steps in that motivating argument, and look just at the repeated premisses at the top of each branch of the infinitely spreading tree! We only have to mention repeated premisses once. So a bit of radical compression will leave us with a nice *finitary* rule:

$$\dfrac{\psi(0) \qquad \forall x(\psi(x) \to \psi(Sx))}{\forall x\psi(x)}$$

This inference rule is evidently sound. And, we can evidently generalize the line of thought, giving us

Defn. 20. The Induction Rule. *For any $\varphi(\xi)$ which expresses a genuine numerical property, the following inference rule is sound*

$$\dfrac{\varphi(0) \qquad \forall x(\varphi(x) \to \varphi(Sx))}{\forall x\varphi(x)}$$

Add this kind of finitary rule to Q [though we'll need to specify more precisely what wffs $\varphi(\xi)$ can feature in the applications of the rule] and – although it isn't as powerful as the infinitary ω-rule – it will at least should plug the particular gap we found in Q, and many more.

4.1.3 Induction: the basic idea

The basic idea of informal mathematics which is reflected in that formal Induction Rule is as follows.

> Whatever numerical property we take, if we can show that (i) zero has that property, and also show that (ii) this property is always passed down from a number n to the next Sn, then this is enough to show (iii) the property is passed down to *all* numbers.

This is the key informal **principle of (arithmetical) induction**, and is a standard method of proof for establishing arithmetical generalizations. It is plainly a sound rule. If a property is passed down from zero from one number to the next, it must percolate down to any given number – since you can get to any number by starting from zero and adding one. (In *IGT2*, we introduce induction as a directly compelling principle, without going via the ω-rule: I thought I'd just ring the changes a bit in these notes!)

For those not so familiar with this standard procedure for establishing arithmetical generalizations, let's have a baby example of the principle at work in an everyday informal mathematical context. We'll take things in plodding detail.

Suppose we want to establish that the sum of the first n numbers is $n(n+1)/2$. Well, first introduce the snappy notation $\chi(n)$ so that

$$\chi(n) \leftrightarrow_{\text{def}} 1 + 2 + 3 + \ldots + n = n(n+1)/2$$

Then (i) trivially $\chi(0)$ is true (the sum of the first zero numbers is zero)! And now suppose that $\chi(n)$ is true, i.e. suppose that $1 + 2 + 3 \ldots n = n(n+1)/2$. Then it follows that

$$\begin{aligned} 1 + 2 + 3 + \ldots + Sn &= (1 + 2 + 3 + \ldots + n) + (n + 1) \\ &= n(n+1)/2 + (n+1) \\ &= (n+1)(n+2)/2 \\ &= (Sn)(Sn+1)/2 \end{aligned}$$

Which means that $\chi(Sn)$ will be true too. Generalizing, then, (ii) for all numbers n, if $\chi(n)$, then $\chi(Sn)$

Given (i) and (ii), by induction we can conclude $\forall n \chi(n)$ – i.e., as we claimed, the sum of the first n numbers is $n(n+1)/2$.

Here's another example of same principle at work, in telegraphic form. Suppose we want to show that all the theorems of a certain Hilbert-style axiomatized

propositional calculus are tautologies. This time, define $\chi(n)$ to be true if the conclusions of proofs appealing only to logical axioms and up to n steps long are tautologies. Then we show that $\chi(0)$ is true (trivial!), and then argue that if $\chi(n)$ then $\chi(Sn)$ (e.g. we note that the last step of an $n+1$-step proof must either be another instance of an axiom, or follow by modus ponens from two earlier conclusions which – since $\chi(n)$ is true – must themselves be tautologies, and either way we get another tautology). Then 'by induction on the length of proofs' we get the desired result.

For a few more examples, see *IGT2*, Ch. 9.

A word to worried philosophers Beginning philosophers, in week one of their first year logic course, have the contrast between deductive and inductive arguments dinned into them. So emphatically is the distinction made, so firmly are they drilled to distinguish conclusive deductive argument from merely probabilistic inductions, that some students can't help feeling initially pretty uncomfortable when they first hear of 'induction' being used in arithmetic!

So let's be clear. We have a case of *empirical, non-conclusive, induction*, when we start from facts about a limited sample and infer a claim about the whole population. We sample some swans, for example, and run k checks showing that $\varphi(0)$, $\varphi(1)$, $\varphi(2)$, ..., $\varphi(k)$ are all true (where $\varphi(n)$ says that swan $\#n$ is white). We hope that these are enough to be representative of the whole population of swans, and so – taking a chance – we infer that for all n, $\varphi(n)$, now quantifying over over all (numbers for) swans, jumping beyond the sample of size k. The gap between the sample and the whole population, between the particular bits of evidence and the universal conclusion, allows space for error. The inference isn't deductively watertight.

By contrast, in the case of *arithmetical induction*, we start not from a bunch of claims about particular numbers but from an already universally quantified claim about all numbers, i.e. $\forall \mathsf{x}(\varphi(\mathsf{x}) \rightarrow \varphi(\mathsf{Sx}))$. We put that universal claim together with the particular claim $\varphi(0)$ to derive another universal claim, $\forall \mathsf{x}\varphi(\mathsf{x})$. This time, then, we are going from universal to universal, and there is no deductive gap.

You might say 'Pity, then, that we use the same word in talking of empirical induction and arithmetical induction when they are such different kinds of inference.' True enough!

4.2 The induction axiom, the induction rule, the induction schema

The basic idea, we said in §4.1.3, is that for any property of numbers, if zero has it and it is passed from one number to the next, then all numbers have it. This intuitive principle is a generalization over properties of numbers. Hence to frame a corresponding formal version, it might seem that we should ideally use a formalized language that enables us to generalize not just over numbers but over properties of numbers. In a phrase, we'd ideally need to be working in a **second-order** theory, which allows second order quantifiers – i.e. we have not only first-order quantifiers running over numbers, but also a further sort of quantifier which runs over arbitrary-properties-of-numbers. Then we could state a second-order

Defn. 21. Induction Axiom.

$$\forall \mathsf{X}([\mathsf{X0} \wedge \forall \mathsf{x}(\mathsf{Xx} \rightarrow \mathsf{XSx})] \rightarrow \forall \mathsf{x}\mathsf{Xx})$$

(Predicates are conventionally written upper case: so too for variables that are to occupy predicate position.)

Despite that, however, we'll concentrate for now on formal theories whose logical apparatus involves only regular first-order quantification. Note: this isn't due to some perverse desire to fight with one hand tied behind our backs. There are some troublesome questions about second-order logic. For a start, there are technical issues: second-order consequence (at least on the natural understanding) isn't effectively axiomatizable – i.e. you can't wrap up the consequence relation for second-order sentences into a nice formalizable logical system. And then there are more philosophical issues: how well do we really understand the idea of quantifying over 'all arbitrary properties of numbers'? Is that a determinate totality which we can quantify over? But we can't go into these worries here and now.

However, if we don't have second-order quantifiers available to range over properties of numbers, how can we handle induction? Well, one way is to adopt the *induction rule* we encountered in §4.1.2:

Defn. 20. The Induction Rule. *For any $\varphi(\xi)$ which expresses a genuine numerical property, the following inference rule is sound*

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

Alternatively, we can set down

Defn. 22. The Induction Schema. *For any $\varphi(\xi)$ which expresses a genuine numerical property, the corresponding instance of this schema*

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

can be taken as an axiom.

Evidently, once we've fixed which predicates $\varphi(\xi)$ express kosher properties, having the *Rule* and having all instances of the *Schema* come to just the same.

Techie note. Strictly speaking, we'll also want to allow uses of the inference rule where φ has slots for additional variables dangling free. Equivalently, we will take the axioms to be the universal closures of instances of the induction schema with free variables. For more explanation, see *IGT2*, §§9.3 and 12.1. We won't fuss about elaborating this point here.

4.3 First-order Peano Arithmetic

4.3.1 Getting generous with induction

Suppose then we start again from Q, and aim to build a richer theory in the language L_A (as defined in §3.4.1) by adding induction.

Any instance of the Induction Schema, we said, should 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. Well, consider *any* open wff φ of L_A. 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 Induction Schema?*

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

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

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) $\forall \mathsf{x}(\varphi(\mathsf{x}) \rightarrow \varphi(\mathsf{Sx}))$. But if we can

already prove things like (i) and (ii) then aren't we already committed to treating φ as a respectable predicate? For given (i) and (ii), 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.

(Techie note: the argument generalizes in the obvious way to the case where $\varphi(\mathsf{x})$ has other variables dangling free. Philosophical note: do you think this positive argument for inductive generosity is compelling?)

4.3.2 Introducing PA

Suppose then that we do agree 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 technically possible intermediate theories, to adopt the much richer theory of arithmetic that we can briskly define as follows:

Defn. 23. PA – **First-order Peano Arithmetic**[1] – *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 that can be constructed from open wffs of L_A.*

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.

First, though, a quick observation. PA allows, in particular, induction for the formula

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

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

But now note that the corresponding $\varphi(0)$ is a trivial logical theorem. Likewise, $\forall x\varphi(Sx)$ is an equally trivial theorem, and that trivially entails $\forall x(\varphi(x) \rightarrow \varphi(Sx))$. So we can use an instance of the Induction Schema inside PA to derive $\forall x\varphi(x)$. But that's just Axiom 3 of Q. 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 ...

4.3.3 Summary overview of PA

First, to repeat, 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 deductive *proof system* is some standard 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. It's convenient, however, to fix officially on a Hilbert-style axiomatic system for later metalogical work theorizing about the theory.

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

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

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

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

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

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

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

plus every 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 of* L_A *that has* 'x' *free.* (Techie note: if $\varphi(x)$ has other variables free then we'll need to 'bind' this instance with universal quantifiers if we want every axiom to be a closed sentence.)

4.4 What PA can prove

PA proves $\forall x(x \neq Sx)$. Just take $\varphi(x)$ to be $x \neq Sx$. Then PA trivially proves $\varphi(0)$ because that's Axiom 1. And PA also proves $\forall x(\varphi(x) \rightarrow \varphi(Sx))$ by contraposing Axiom 2. And then an induction axiom 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. no number is a self-successor. It's as simple as that. Yet this trivial little result is worth noting when we recall our deviant interpretation which makes the axioms of Q true while making $\forall x(0 + x = x)$ false: that interpretation featured Kurt Gödel himself added to the domain as a rogue self-successor. A bit of induction, however, rules out self-successors.

And so it goes: the familiar basic truths about elementary general truths about the successor function, addition, multiplication and ordering (with the order relation as defined in §3.6) are all provable in PA using induction – and these theorems rule out other simple deviant models. There are more than enough examples worked through in *IGT2*, which we won't repeat here!

So we might reasonably have hoped – at least before we'd heard of Gödel's incompleteness theorems – that PA would turn out to be a complete theory that indeed pins down all the truths of L_A.

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 the language 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 – the argument uses a standard model-theoretic method called 'elimination of quantifiers' which isn't hard, and was known in the 1920s, but it would just take too long to explain.

So the situation is as follows, and was known before Gödel got to work. (i) There is a complete formal axiomatized theory BA whose theorems are exactly the quantifier-free truths expressible using successor, addition and multiplication (and the connectives). (ii) There is another complete formal axiomatized 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 wasn't obviously predictable that multiplication would make all the difference. Yet it does. In fact, as we've said before, as soon we have an

arithmetic as strong as Q, we get incompleteness.

And by the way, it isn't that a theory of multiplication must in itself be incomplete. In 1929, 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. So why then does putting multiplication together with addition and successor produce incompleteness? The answer will emerge shortly enough, but pivots on the fact that an arithmetic with all three functions built in can express/capture *all* 'primitive recursive' functions. But we'll have to wait to the next-but-one episode to explain what that means.

4.5 Burning questions

1. PA *has an infinite number of axioms: why is having an infinite number of axioms any better than using an infinitary* ω*-rule?* We've already answered this in passing. Sure, there is an unlimited number of instances of the induction schema, each one of which is an axiom of PA. Still, we can mechanically check a given wff to see whether it is or isn't one of the instances. So we can mechanically check a wff to see whether it is a PA axiom. So we can mechanically (and finitely) check a given finite array of wffs to see whether it is a properly constructed PA proof. By contrast, we obviously can't finitely check an array to see if it involves a correct application of the infinite-premiss ω-rule. That's why PA is a kosher effectively axiomatized formal theory in our official sense, and a system with the ω-rule isn't.

2. *But won't some of the instances of the induction schema be ludicrously long, far too long to mechanically check?* Ah, but remember we are talking about checkability in principle, without constraints on time, the amount of ink to be spilt, etc. etc. Effective decidability is not practical decidability.

3. PA *has an infinite number of axioms: but can we find a finite bunch of axioms with the same consequences?* No. First-order Peano Arithmetic is essentially infinitely axiomatized (not an easy result though!).

4. *We saw that* Q *has 'a non-standard model', i.e. there is a deviant unintended interpretation that still makes the axioms of* Q *true. Does* PA *have any non-standard models, i.e. deviant unintended interpretations that still make* its *axioms true?* Yes, by the Löwenheim-Skolem theorem it must do (because any consistent theory with infinite models will have models of arbitrary infinite size). True, PA does not have the trivial non-standard

model we built for Q. But PA still doesn't pin down uniquely the structure of the natural numbers. Indeed, even if we assume that we are looking for a countable model – i.e. a model whose elements could in principle be numbered off – there can be non-standard models of PA. A standard compactness argument shows this. Here it is:

A compactness argument If you know the compactness theorem for first-order theories, you'll probably have met the following proof.

Suppose we add to the language of PA a new constant c, and add to the axioms of PA the additional axioms $c \neq 0$, $c \neq \overline{1}$, $c \neq \overline{2}$, ..., $c \neq \overline{n}$, ... Evidently each finite subset of the axioms of the new theory has a model (assuming PA is consistent and has a model): just take the standard model of arithmetic and interpret c to be greater than the maximum n for which $c \neq \overline{n}$ is in the given finite suite of axioms.

Now apply the compactness theorem. Since each finite subset of the infinite set of axioms of the new theory has a model, so does the whole theory. And now, by the downward Löwenheim-Skolem theorem there will be a countable model of this theory, which contains a zero, its successors-according-to-the-theory, and rogue elements including the denotation of c. Since this structure is a countable model of PA-plus-some-extra-axioms it is, a fortiori, a countable model of PA, and must be distinct from the standard one as its domain has more than just the zero and its successors.

'OK: so can you describe one of these countable-but-weird models of PA? Being a countable model, we can take its domain to be the numbers again: but what do the interpretations of the successor, addition and multiplication functions now look like?' Well, it is difficult to say, for a principled reason. **Tennenbaum's Theorem** tells us that, for any non-standard model of PA, the interpretations of the addition and the multiplication functions can't be nice computable functions, and so can't be arithmetical functions that we can give a familiar sort of description of. That indeed makes it more difficult to get a direct handle on the non-standard models.

Further reading

Read *IGT2*, Ch. 9, §12.1, Ch. 13.

For more on induction, see e.g. Daniel J. Velleman, *How to Prove It* (CUP, 1994), Ch. 6. Also see http://en.wikipedia.org/wiki/Mathematical_induction.

For more on Peano Arithmetic see http://en.wikipedia.org/wiki/Peano_axioms.And for another book presentation of PA, see e.g. the classic Elliott Mendelson, *Introduction to Mathematical Logic*, Ch. 3 §1. (NB. Both these versions take the identity laws as part of the arithmetic rather than as part of the background logic.)

Enthusiasts can find out more about Tennenbaum's Theorem (and even how to prove it) here: http://www.logicmatters.net/resources/pdfs/TennenbaumTheorem.pdf.

Episode 5

Quantifier complexity

- The idea of Δ_0, Σ_1, and Π_1 wffs
- Addendum: A consistent extension of Q is sound for Π_1 wffs

---

PA is the canonical, most natural, first-order theory of the arithmetic of successor, addition and multiplication. Indeed it is arguable that *any* proposition about successor, addition and multiplication that can be seen to be true just on the basis of our grasp of the structure of the natural numbers, without more sophisticated 'higher-order' reflections, can be shown to be true in PA (for discussion of this, see *IGT2*, §30.4). Still there is some formal interest in exploring weaker systems, sitting between Q and PA, systems which have *some* induction, but not induction for all open L_A wffs. For example, there is some interest in the theories you get by allowing as axioms only instances of the induction schema induction for so-called Δ_0 wffs, or so-called Σ_1 wffs. We pause over this in *IGT2*, Ch. 12.

However, we are not going to explore these weak arithmetics in these present notes. But, irrespective of that, you really need to know what Δ_0 , Σ_1, and Π_1 wffs are. Later, for example, we'll find that Σ_1 wffs suffice to capture decidable properties of numbers, and that the standard Gödel sentence that sort-of-says "I am unprovable" is a Π_1 wff. So this unusually short episode briskly explains what this sort of claim means.

5.1 Q knows about bounded quantifications

Reread §3.6 of these notes, where we explain how to add $\leq$ to the language of basic arithmetic

Now, we often want to say that all/some numbers less than or equal to some bound have a particular property. We can now express such claims in formal arithmetics like Q and PA using wffs of the shape $\forall\xi(\xi \leq \kappa \rightarrow \varphi(\xi))$ and $\exists\xi(\xi \leq \kappa \wedge \varphi(\xi))$, where $\xi \leq \zeta$ is just short for $\exists\mathsf{v}(\mathsf{v} + \xi = \zeta)$. Here κ can stand in for any term (not just numerals), so long as it doesn't contain ξ free. And it is standard to further abbreviate such wffs by $(\forall\xi \leq \kappa)\varphi(\xi)$ and $(\exists\xi \leq \kappa)\varphi(\xi)$ respectively.

Now note that we have results like these:

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

Results like these show that Q – and hence a stronger theory like PA – 'knows' that bounded universal quantifications (with fixed number bounds) behave like finite conjunctions, and that bounded existential quantifications (with fixed number bounds) behave like finite disjunctions. Hold on to that thought!

5.2 Δ_0, Σ_1 and Π_1 wffs

Let's informally say that

Defn. 24. *An L_A wff is Δ_0 if its only quantifications are bounded ones.*

For a fancied-up definition, see *IGT2*, §11.4. So a Δ_0 wff (some say Σ_0) is one which is built up using the successor, addition, and multiplication functions, identity, the less-than-or-equal-to relation (defined as usual), plus the familiar propositional connectives and/or *bounded* quantifications.

In other words, a Δ_0 wff is just like a quantifier-free L_A wff, i.e. like an L_B wff, except that we in effect allow ourselves to wrap up some finite conjunctions into bounded universal quantifications, and similarly wrap up some finite disjunctions into bounded existential quantifications.

Now, we can mechanically calculate the truth-value of every quantifier-free L_A sentence, i.e. L_B sentence. And a Δ_0 sentence is tantamount to a quantifier-free sentence. (Why? Well, note that in a Δ_0 *sentence*, the bound of the outermost quantifier can have no variables in it. Therefore Q proves that this bound is equal to a particular numeral, and so the entire quantification is equivalent to a finite conjunction or disjunction of simpler Δ_0 *sentences*, which we can then apply the same process to in turn to get rid of all the bounded quantifiers.) It should follow, then, that we can mechanically calculate the truth-value of any Δ_0 sentence. And it indeed does follow: that's Theorem 11.2 in *IGT2*!

A corollary is that we can mechanically determine whether a Δ_0 open wff $\varphi(\mathsf{x})$ is satisfied by a number n by determining whether $\varphi(\overline{\mathsf{n}})$ is true (likewise for open wffs with more than one free variable). So, Δ_0 open wffs express decidable properties of numbers.

Since we know from Theorem 11 that even Q can correctly decide all quantifier-free L_A sentences, and Q knows that *bounded* quantifications with numeral bounds behave just like conjunctions/disjunctions, it won't be a surprise to hear that we have

Theorem 16. Q *(and hence* PA*) can correctly decide all* Δ_0 *sentences.*

But again, we won't give the proof here: the details are spelt out in establishing Theorem 11.2 in *IGT2*.

We next say, again informally, that

Defn. 25. *An* L_A *wff is* Σ_1 *if it is (or is logically equivalent to) a* Δ_0 *wff preceded by zero, one, or more unbounded* existential *quantifiers. And a wff is* Π_1 *if it is (or is logically equivalent to) a* Δ_0 *wff preceded by zero, one, or more unbounded* universal *quantifiers.*

As a mnemonic, 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.

So a Σ_1 wff says that some number (pair of numbers, etc.) satisfies the decidable condition expressed by its Δ_0 core; likewise a Π_1 wff says that every number (pair of numbers, etc.) satisfies the decidable condition expressed by its Δ_0 core.

To check understanding, make sure you understand why

1. The negation of a Δ_0 wff is Δ_0.

2. A Δ_0 wff is also Σ_1 and Π_1.

3. The negation of a Σ_1 wff is Π_1.

Now we can in fact keep on going, to consider wffs with greater and greater quantifier complexity. So, we say a Π_2 wff is (or is logically equivalent to) 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. Likewise, a Σ_2 wff is (equivalent to) one that starts with two blocks of quantifiers, a block of existential quantifiers followed by a block of universal quantifiers followed by a bounded kernel. And so it goes, up the so-called **arithmetical hierarchy** of increasing quantifier complexity. But we won't need to consider higher up that the first levels of the arithmetical hierarchy here (though higher levels are significant in the general theory of computation).

Instead, we finish this section with an easy result:

Theorem 17. Q *(and hence* PA*) can prove any true* Σ_1 *sentences (is '*Σ_1*-complete').*

Proof. Take, for example, a 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. But then by Theorem 16, 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 truths which are (or are provably-in-Q equivalent to) some Δ_0 wff preceded by one or more unbounded existential quantifiers. □

A remarkable corollary But if our last theorem is entirely straightforward and unexciting, it has an immediate corollary which is interesting and perhaps initially surprising:

Theorem 18. *If* T *is a consistent theory which includes* Q*, then every* Π_1 *sentence that it proves is true.*

Proof. Suppose T proves a *false* Π_1 sentence φ. Then $\neg\varphi$ will be a *true* Σ_1 sentence. But in that case, since T includes Q and so is 'Σ_1-complete', T will prove $\neg\varphi$, making T inconsistent. Contraposing, if T is consistent, any Π_1 sentence it proves is true. □

Which is, in its way, a 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.

For example, it turns out that, with some trickery, we can state Fermat's Last Theorem as a Π_1 sentence. And famously, Andrew Wiles has shown how to prove Fermat's Last Theorem using some very heavy-duty infinitary mathematics. Now we see, rather intriguingly, that we actually don't have to believe that this infinitary mathematics is *true* – whatever exactly that means when things get so very wildly infinitary! – but only that it is *consistent*, to take him as establishing that the Π_1 arithmetical claim which is the Theorem is true.

Further reading

Now read *IGT2*, Ch. 11 – and then go on to read Ch. 12 and re-read Ch. 13 as well. See also the brief http://en.wikipedia.org/wiki/Bounded_quantifier and glance at (but don't get fazed by) http://en.wikipedia.org/wiki/Arithmetical_hierarchy.

Enthusiasts who want to learn something (in very general terms) about arithmetics – meaning, as always, theories of arithmetic – which are between Q and PA in strength will find all kinds of information in this short blogpost (and browse too through the comments on it): http://golem.ph.utexas.edu/category/2011/10/weak_systems

Episode 6

Primitive recursive functions

- What's a primitive recursive function?
- How to prove results about all p.r. functions
- The p.r. functions are computable ...
- ... but not all computable functions are p.r.
- The idea of a characteristic function, which enables us to define ...
- ... the idea of p.r. properties and relations.

In our preamble, it might be helpful this time to give a story about where we are going, rather than review again where we've been. So, at the risk of spoiling the excitement, here's what's going to happen in this and the following three episodes.

1. 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 stop there? Even school arithmetic acknowledges many more numerical functions, like the factorial and the exponential.

This episode describes a very wide class of familiar such functions, the so-called primitive recursive ones. They are a major subclass of the effectively computable functions.

We also define the primitive recursive properties and relations (i.e. those with a p.r. 'characteristic function') – a p.r. function can effectively decide when the property/relation holds.

2. Episode 7 then shows that L_A, the language of basic arithmetic (see §3.4.1), can *express* all p.r. functions and relations. Moreover Q and hence PA can *capture* all those functions and relations too (i.e. case-by-case prove wffs that assign the right values to the functions for particular numerical arguments). So Q and PA, despite having only successor, addition and multiplication 'built in', can actually deal with a vast range of functions (at least in so far as they can 'calculate' the value of the functions for arbitrary numerical inputs).

   Note, by the way, the link with our earlier talk about "sufficiently strong theories" see Defn. 17). Those, recall, are theories that can capture *all* effectively decidable properties of numbers. Well, now we are going to show that PA (indeed, even Q) can capture at all least all decidable-by-a-p.r.-function properties of numbers. And that's enough for the Gödelian argument to go through as we well see.

3. In Epsiode 8 we introduce again the key idea of the 'arithmetization of syntax' by Gödel-numbering which we first met in §1.7. Focus on PA for the moment. Then we can define various properties/relations as follows:

   $Wff(n)$ iff n is the code number of a PA-wff;
   $Sent(n)$ iff n is the code number of a PA-sentence;
   $Prf(m, n)$ iff m is the code number of a PA-proof of the sentence with code number n.

   Moreover – the crucial result – these properties/relations are primitive recursive. Similar results obtain for any sensibly axiomatized formal theory.

4. Since Prf is p.r., and PA can capture all p.r. relations, there is a wff $\mathsf{Prf}(\mathsf{x}, \mathsf{y})$ which captures the relation Prf. In Episode 9 we use this fact – or a closely related one – to construct a Gödel sentence which sort-of-says 'I am not provable in PA', and hence prove Gödel first incompleteness theorem for PA. Similarly for other sensibly axiomatized arithmetics that include Q.

Now read on ...

6.1 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** function, 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. The equations determine, e.g., that $3^4 = 3 \times 3 \times 3 \times 3 = 81$.

We've seen this two-clause pattern before, of course, in our formal Axioms in Q/PA for the addition and multiplication functions. Presented in the style of everyday informal mathematics (leaving quantifiers to be understood) – and note, everything in this episode *is* just informal mathematics – 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. A procedure where we evaluate a function for one input by calling the *same* function for a smaller input or inputs is standardly termed 'recursive' –

and the particularly simple pattern we've illustrated is called, more precisely, 'primitive recursive'. So 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 examples so far can be put together to illustrate 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 second 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

Defn. 26. *Roughly: 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 in 1888.

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

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.

6.2 Defining the p.r. functions more carefully

On the one hand, I suppose you ought to read this section! On the other hand, *don't* get lost in the techie details. All we are trying to do here is give a careful, explicit, presentation of the ideas we've just been sketching, and flesh out that rough and ready Defn. 26.

6.2.1 More on definition by primitive recursion

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.

6.2.2 The initial functions

With that motivation, we will now officially declare the full 'starter pack' of functions as follows:

Defn. 27. *The* **initial functions** *are the successor function S, the zero function $Z(x) = 0$ and all the k-place identity functions, $I^k_i(x_1, x_2, \ldots, x_k) = x_i$ for each k, and for each i, $1 \leq i \leq k$.*

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.

6.2.3 Primitive recursion and by composition: the official story

We'll now, just for the record, 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 $x_1, x_2, \ldots, x_k$ (take k to be fixed by context). Then we can generalize as follows:

Defn. 28. *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.

We also, just for the record, 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:

Defn. 29. *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 $h(g(x, y), g'(y, z))$. [No one promised that these details were going to be exciting!]

6.2.4 Putting everything together

We informally defined the primitive recursive (henceforth, p.r.)[3] 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 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**:

Defn. 30. *The p.r. functions are as follows*

[3] Terminology alert: some authors writing in this area use 'p.r.' as short for *partial* recursive – a different notion!

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.

So: 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. Let's say:

Defn. 31. *A* **full definition** *for the p.r. function f is a specification of a sequence of functions $f_0, f_1, f_2, \ldots, f_k$ where each f_j is either an initial function or is defined from previous functions in the sequence by composition or recursion, and $f_k = f$.*

Then what we've seen is that every p.r. function is required to have a full definition in this defined sense. (That's the sharp version of the informal characterization we gave at the end of §6.1.)

6.3 How to prove a result about all p.r. functions

That last point that every p.r. function has a full definition means that there is a simple method of proving that every p.r. function has some property P (the method we've in fact just used to show that all p.r. functions are total). For suppose that, for some given property P, we can show

P1. The initial functions have property P.

P2. If the functions g and h have property P, and f is defined by composition from g and h, then f also has property P.

P3. If the functions g and h have property P, and f is defined by primitive recursion from g and h, then f also has property P.

Then P1, P2, and P3 together suffice to establish that all primitive recursive functions have property P.

Why so? Well, as we said, any p.r. function f has a *full* definition, which specifies a sequence of functions $f_0, f_1, f_2, \ldots, f_k$ where each f_j is either an initial function or is defined from previous functions in the sequence by composition or recursion, and $f_k = f$. So as we trek along the f_j, we start with initial functions which have property P by P1. By P2 and P3, each successive definitional move takes us from functions which have property P to another function with property P. So, every function we define as we go along has property P, including the final target function f. (This proof is, in effect, a proof by induction on the length of the chosen full definition for f: do you see why? See above, §4.1.3.)

In sum, then: to prove that all p.r. functions have some property P, it suffices to prove the relevant versions of P1, P2 and P3.

6.4 The p.r. functions are computable

6.4.1 The basic argument

We want to show that every p.r. function is mechanically computable. Given the general strategy just described, it is enough to show that

C1. The initial functions are computable.

C2. If f is defined by composition from computable functions g and h, then f is also computable.

C3. If f is defined by primitive recursion from the computable functions g and h, then f is also computable.

But C1 is trivial: the initial functions S, Z, and I_i^k are effectively computable by a simple algorithm. And C2, 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).

To illustrate C3, 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. And obviously the argument generalizes.

6.4.2 Computing p.r. functions by 'for'-loops

Compare our p.r. definition of the factorial 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 this kind of bounded **'for' loop** is that the total number of iterations to be run through is bounded in advance: we number the loops from 0, and in executing the loop, you increment the counter by one on each cycle until you hit the bound. So in this case, 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 bounded '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 $\Rightarrow$ multiplication $\Rightarrow$ addition $\Rightarrow$ 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, giving us

Theorem 19. *Primitive recursive functions are effectively computable by a series of (possibly nested) bounded 'for' loops.*

6.4.3 If you can compute it using 'for'-loops, it is p.r.

The converse is also true. Take a 'for' loop which computes the value of a one-place function $f(n)$, a loop which calls on two prior routines, one which computes a constant g (used to set the value of $f(0)$), the other which computes a function h (which is used on the $n + 1$-th loop to fix the value of $f(Sn)$ in terms of the value of $f(n)$). This plainly corresponds to a definition by recursion of f in terms of g and h. And generalizing,

Theorem 20. *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.

6.5 Not all computable numerical functions are p.r.

We have seen that any p.r. function is mechanically computable. And most of the ordinary computable numerical functions you already know about from elementary maths are in fact primitive recursive. *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.

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 'for' loops as its main programming structure. Each loop goes through a specified number of iterations. However, we elsewhere do allow computations to involve *open-ended searches*, with no prior bound on the length of search. We made essential use of this permission 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.*[4]

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 21. *There are effectively computable numerical functions which aren't primitive recursive.*

Proof. 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.

[4] A complication: I'm told that in a language in the syntactic tradition of C (the likes of Javascript, C#, PHP, ...), a 'for' loop is actually just a 'while' loop in disguise and is not restricted to looping a precomputed number of times. So I guess I've been thinking all along of 'for' loops as in Old School languages like Basic or Pascal. Showing my age here!

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

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

Now 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,[5] by putting $\delta(n) = f_n(n) + 1$. 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.

[5] Compare the inset box on diagonalization on p. 23.

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.[6]

Can we effectively list all computable numerical functions? The last argument evidently generalizes.

Suppose we can effectively list the recipes for some class C of computable total (i.e. everywhere-defined) functions from natural numbers to natural numbers, where the algorithms compute in turn the functions f_0^C, f_1^C, f_2^C, ... Then again we can define $\delta^C(n) = f_n^C(n) + 1$. This will be a total function, by assumption because f_n^C is everywhere defined, so defined for input n in particular. Again $\delta^C(n) + 1$ is computable — just go along our effectively generated list of algorithms until to yet to the n-th one, and then run that algorithm on input n. But, as before δ^C cannot be one of the f_j^C. In a slogan, we can 'diagonalize out' of class C, and get another computable function.

So C can't contain all the computable (one=place, numerical) functions. That gives us a theorem:

Theorem 22. *No effective listing of algorithms can include algorithms for* all *the intuitively computable total (one place, numerical) functions.*

Note that the restriction to total functions is doing essential work here. Consider algorithms for *partial* computable functions (the idea is that when the algorithm for the partial function φ_i 'crashes' on input n, $\varphi_i(n)$ is undefined). And consider a listing of algorithms for partial functions. Then the

[6] To expand that thought a bit, note that in the algorithm to compute a p.r. function, the nesting depth of the for-loops is fixed. But in order to compute the diagonal function δ we have to be able to evaluate in turn the n-th p.r. function for the input n, and as we go down the list we get functions whose algorithms will have loops of varying depths – so our computation of $\delta(n)$ will involve going through a nest of loops of varying depth depending on the input n. (I owe that observation to Henning Makholm.)

diagonal function $\delta(n) = \varphi_n(n) + 1$ could then consistently appear on the list e.g. as φ_d, if $\varphi_d(d)$ and hence $\varphi_d(d) + 1$ are both undefined.

6.6 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. decidable (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.

Defn. 32. *The characteristic function of the numerical property P is the one-place function c_P such that if m is P, then $c_P(m) = 0$, and if m isn't P, then $c_P(m) = 1$.*

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

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

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

Defn. 33. *A* **p.r. decidable property** *is a property with a p.r. characteristic function, and likewise a* **p.r. decidable relation** *is a relation with a p.r. characteristic function.*

Check you see why, given that any p.r. function is effectively computable, p.r. decidable properties and relations are indeed among the effectively decidable ones? By way of causal abbreviation, we'll fall into saying that p.r. decidable properties and relations are (simply) p.r.

Further reading

Read *IGT2*, Ch. 14.

There's a sound Wikipedia entry at http://en.wikipedia.org/wiki/Primitive_recursive.

There's an early classic treatment of the notion of a primitive recursive function in Stephen C. Kleene's *Introduction to Metamathematics*, (North-Holland, 1952), Ch. IX. (read the first three or four sections, then enthusiasts can go through to the end of the chapter). For a much more recent and very accessible treatment, try Richard L. Epstein and Walter Carnielli's *Computability: Computable Functions, Logic, and the Foundations of Mathematics* (Wadsworth, 2000; Advanced Reasoning Forum, 2008), Ch. 11.

Episode 7

Expressing and capturing the primitive recursive functions

- L_A can express all p.r. functions
- The role of the β-function trick in proving that result
- In fact, Σ_1 wffs suffice to express all p.r. functions
- Q can capture all p.r. functions
- Expressing and capturing p.r. properties and relations.

The last episode wasn't about logic or *formal* theories at all: it was about common-or-garden arithmetic and the informal notion of computability.

We noted that addition can be defined in terms of repeated applications of the successor function. Multiplication can be defined in terms of repeated applications of addition. The exponential and factorial functions can be defined, in different ways, in terms of repeated applications of multiplication. There's already a pattern emerging here! And the main task in the last episode was to get clear about this pattern.

So first we said more about the idea of defining one function in terms of repeated applications of another function. Tidied up, that becomes the idea of *defining a function by primitive recursion* (Defn. 28).

Then we want the idea of a definitional chain where we define a function by primitive recursion from other functions which we define by primitive recursion from other functions, and so on down, until we bottom out with the successor function and other trivia. We also of course allow composition of functions – i.e. feeding the output of one already-defined function into another already-defined function – along the way. Tidied up, this gives us the idea of a *primitive recursive function*, i.e. one that can be defined by such a definitional chain (Defn. 30).

We then noted three key facts:

1. Every p.r. function is intuitively computable – moreover it is computable using only 'for' loops and not open-ended 'do until' loops. That's Theorem 19.

2. Conversely, if a numerical function can be computed from a starter pack of trivial functions using only 'for' loops, then it is primitive recursive. That's Theorem 20.

3. But not every intuitively computable numerical function is primitive recursive. That's Theorem 21.

OK, so the situation is now this. In Episodes 3 and 4, we introduced some *formal* arithmetics with just three functions – successor, addition, multiplication – built in. But we have now reminded ourselves that ordinary *informal* arithmetic talks about many more elementary functions like the exponential, the factorial, and so on and so forth: and we generalized the sort of way these functions can be defined to specify the whole class of primitive recursive functions. A gulf seems to have opened up between the modesty of the resources of our formal theories (including the strongest so far, PA) and the richness of the world of p.r. functions (and we know that those aren't even all the computable arithmetical functions). In this episode, we show the gulf is merely apparent. **The language L_A in fact can *express* all p.r. functions; and even the weak theory Q can *capture* them all too (using Σ_1 wffs).** So, in fact, our formal theories – despite their modest basic resources – can deal with a lot more than you might at first sight suppose.

Now recall the idea of a sufficiently strong theory which we introduced in Defn. 17. That was the idea of capturing all decidable numerical properties. That's equivalent to the idea of capturing all computable one-place functions

(by the link between properties and their characteristic functions). Well, what we are now claiming is that we can show that Q (and hence PA) can at least capture all primitive recursive computable functions. That will be enough for Gödel's argument for incompleteness to fly.

A very import remark, though, before we set off. You might well think that some of the details in this episode – particularly about the so-called β-function trick – are a bit messy. In fact, the ideas here are not difficult. But still, don't worry at all if you find it tedious/tricky to get your head round this kind of thing. **Nothing in what follows in later episodes depends on your knowing the details of the *proofs* of the Theorems below about expressing/capturing functions. All you need to know for further applications is that the results announced in bold above *can* be proved.** So please don't let this episode be a stumbling block to continuing further in the these notes. The proofs are outlined here for those who want to know how the results are established: but do feel free to lightly skim and skip on to the next episode.

7.1 L_A can express the factorial function

In this section we are going to show that L_A – despite having only successor, addition and multiplication built in – can in fact express the factorial function. That is to say, we can construct a wff $\mathsf{F}(\mathsf{x}, \mathsf{y})$ such that $\mathsf{F}(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ if and only if $n = m!$. Then in the next section we'll use the same key trick to show that L_A can express any p.r. function at all.

7.1.1 What we need to express the factorial function

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?

Well, 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 $y = x!$, the p.r. definition comes to this:

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?

What we need to do is to wrap up a finite sequence into a single **code number** c, and then have e.g. a two-place **decoding function** *decode* such that if you give to *decode* as inputs the code c and the index i it then spits out the i-th member of the sequence which c codes! In other words, if c is the code number for the sequence $k_0, k_1, \ldots, k_x$ we want: $decode(c, i) = k_i$.

If we can find a coding scheme and a decoding function that does the trick, then we can rewrite (A) as follows, talking about a code number c instead of the sequence $k_0, k_1, \ldots, k_x$, and writing $decode(c, i)$ instead of k_i:

B. There is a code number c such that: $decode(c, 0) = 1$, and if $u < x$ then $decode(c, Su) = decode(c, u) \times Su$, and $decode(c, x) = y$.

And hence, if a suitable *decode* function can be expressed in L_A, then we can define the factorial in L_A.

7.1.2 Gödel's β-function

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

> A three-place decoding function is a function of the form $decode(c, d, i)$ 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 $i \leq n$, $decode(c, d, i) = k_i$.

A three-place decoding-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 decoding-function in terms of the functions built into basic arithmetic. But Gödel neatly solved our problem with the following little trick. Put

$\beta(c, d, i) =_{\text{def}}$ the remainder left when c 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 (mathematical completists: see *IGT2*, §15.2, fn. 4 for proof that this works!)

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:

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

If you think about it, this expresses our Gödelian β-function in L_A (for remember, we can define '$\leq$' in L_A).

7.1.3 Defining the factorial in L_A

We've just claimed: given any sequence of numbers $k_0, k_1, \ldots, k_x$, there are code numbers c, d such that for $i \leq x$, $\beta(c, d, i) = k_i$. So have $y = x!$ just in case

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 $k_x = y$.

And we can now reformulate this as follows:

B′. There is some pair of code numbers 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 the β-function can be expressed in L_A by the open wff we abbreviated B. So we can render (B′) into L_A as follows:

C. $\exists \mathsf{c} \exists \mathsf{d}\{\mathsf{B(c, d, 0, \overline{1})} \wedge$
$(\forall \mathsf{u} \leq \mathsf{x})[\mathsf{u} \neq \mathsf{x} \rightarrow \exists \mathsf{v} \exists \mathsf{w}\{(\mathsf{B(c, d, u, v)} \wedge \mathsf{B(c, d, Su, w)}) \wedge \mathsf{w} = \mathsf{v} \times \mathsf{Su}\}] \wedge$
$\mathsf{B(c, d, x, y)}\}$.

Abbreviate all that by '$\mathsf{F(x, y)}$', and we've arrived. For this evidently expresses (B′) which is equivalent to (A) and so expresses the factorial function. Neat, eh?

7.2 L_A can express all p.r. functions

We want to generalize, using the β-function trick again, to show that if the one-place function f is p.r., then there is a two-place L_A wff $\varphi(\mathsf{x},\mathsf{y})$, such that $\varphi(\overline{\mathsf{m}},\overline{\mathsf{n}})$ is true if and only if $f(m) = n$. And similarly, of course, for many-place p.r. functions.

Suppose, just suppose, that the following three propositions are all true:

E1. L_A can express the initial functions. (See Defn. 27.)

E2. 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. (See Defn. 29.)

E3. 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. (See Defn. 28.)

Then by the argument of §6.3, those assumptions are enough to establish

Theorem 23. *L_A can express all p.r. functions.*

(Before proceeding any further, make sure you understand why the theorem follows from the assumptions.)

Proof of E1. Just look at cases. The successor function $Sx = y$ is of course expressed by the open wff $\mathsf{Sx} = \mathsf{y}$.

The zero function, $Z(x) = 0$ is expressed by the wff $\mathsf{Z}(\mathsf{x},\mathsf{y}) =_{\text{def}} (\mathsf{x} = \mathsf{x} \wedge \mathsf{y} = \mathsf{0})$.

Finally, the three-place identity function $I^3_2(x,y,z) = y$, to take just one example, is expressed by the wff $\mathsf{I}^3_2(\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{u}) =_{\text{def}} y = u$ (or we could use $(\mathsf{x} = \mathsf{x} \wedge \mathsf{y} = \mathsf{u} \wedge \mathsf{z} = \mathsf{z})$ if we'd like x and z actually to appear in the wff). Likewise for all the other identity functions. [Check those claims!] □

Proof of E2. Suppose g and h are one-place functions, expressed by the wffs $\mathsf{G}(\mathsf{x},\mathsf{y})$ and $\mathsf{H}(\mathsf{x},\mathsf{y})$ respectively. Then, the function $f(x) = h(g(x))$ is evidently expressed by the wff $\exists\mathsf{z}(\mathsf{G}(\mathsf{x},\mathsf{z}) \wedge \mathsf{H}(\mathsf{z},\mathsf{y}))$.

For suppose $g(m) = k$ and $h(k) = n$, so $f(m) = n$. Then by hypothesis $\mathsf{G}(\overline{\mathsf{m}},\overline{\mathsf{k}})$ and $\mathsf{H}(\overline{\mathsf{k}},\overline{\mathsf{n}})$ will be true, and hence $\exists\mathsf{z}(\mathsf{G}(\overline{\mathsf{m}},\mathsf{z}) \wedge \mathsf{H}(\mathsf{z},\overline{\mathsf{n}}))$ is true, as required. Conversely, suppose $\exists\mathsf{z}(\mathsf{G}(\overline{\mathsf{m}},\mathsf{z}) \wedge \mathsf{H}(\mathsf{z},\overline{\mathsf{n}}))$ is true. Then since the quantifiers run over numbers, $(\mathsf{G}(\overline{\mathsf{m}},\overline{\mathsf{k}}) \wedge \mathsf{H}(\overline{\mathsf{k}},\overline{\mathsf{n}}))$ must be true for some k. So we'll have $g(m) = k$ and $h(k) = n$, and hence $f(m) = h(g(m)) = n$ as required.

Other cases where g and/or h are multi-place functions can be handled similarly. □

Proof of E3. The tricky case! 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.

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 L_A by

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

Abbreviate this defined wff $\varphi(\vec{\mathsf{x}}, \mathsf{y}, \mathsf{z})$; it is then evident that φ will serve to express the p.r. defined function f. Which gives us the desired result E3. □

So, we've shown how to establish each of the claims E1, E2 and E3. Hence every p.r. function can be expressed in L_A.

Theorem 23 is in the bag! Phew!!

The proof of Theorem 23 is constructive What we have just showed, in effect, is how to take a chain of definitions by composition and primitive recursion – starting with the initial functions and building up to a full definition for f – and then step-by-step reflect it in building up to a wff that expresses f.

A full definition for f yields a whole sequence of functions $f_0, f_1, f_2, \ldots, f_k$ where each f_j is either an initial function or is constructed out of previous functions by composition or recursion, and $f_k = f$. Corresponding to that sequence of functions we can write down a sequence of L_A wffs which express those functions. We write down the E1 expression corresponding to an initial function. If f_j comes from two previous functions by composition, we use the existential construction in E2 to write down a wff built out of the wffs expressing the two previous functions. If f_j comes from two previous functions by recursion, we use the β-function trick and write down a C*-style expression built out of the wffs expressing the two previous functions.

So that means we've not only proved that for any given p.r. function f *there exists* an L_A-wff which expresses it. We've shown how to *construct* such a wff by recapitulating the structure of a definitional 'history' for f. The proof is, in a good sense, a constructive one.

7.3 Canonical wffs for expressing p.r. functions are Σ_1

For brevity, let's say (in the light of point in the last box) that

Defn. 34. *An L_A wff* canonically expresses *the p.r. function f if it recapitulates a full definition for f by being constructed in the manner described in the proof of Theorem 23.*

There can be other wffs which also express a given p.r. function f: but these canonical ones from which we can read off a full definition f will interest us the most.

Now, a canonical wff which reflects a full definition of f is built up starting from wffs expressing initial wffs. Those starter wffs are Δ_0 wffs, and hence Σ_1.

Suppose g and h are one-place functions, expressed by the Σ_1 wffs $\mathsf{G}(\mathsf{x},\mathsf{y})$ and $\mathsf{H}(\mathsf{x},\mathsf{y})$ respectively. The function $f(x) = h(g(x))$ is expressed by the wff $\exists\mathsf{z}(\mathsf{G}(\mathsf{x},\mathsf{z}) \wedge \mathsf{H}(\mathsf{z},\mathsf{y}))$ which is Σ_1 too. For that is equivalent to a wff with the existential quantifiers pulled from the front of the Σ_1 wffs G and H out to the very front of the new wff. Similarly for other cases of composition.

Finally, if we can already express the one-place function g by a two-variable Σ_1 expression G, and the three-place function h by the *four*-variable Σ_1 expression

H. Then if the one-place f is defined from g and h by primitive recursion, f can be expressed by the wff $\mathsf{F}(\mathsf{x},\mathsf{y})$ defined by

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

And this too is Σ_1. For B is Δ_0: and C* is equivalent to what we get when we drag all the existential quantifiers buried at the front of each of B, G and H to the very front of the wff. (Yes, dragging existentials past a universal is usually wicked! – but here the only universal here is a bounded universal, which is 'really' just a tame conjunction, and simple tricks explained in *IGT*2 allow us to get the existentials all at the front). Again this generalizes to other cases of definition by recursion.

So in fact our recipe for building a canonical wff in fact gives us a Σ_1 wff. Which yields

Theorem 24. *L_A can express any p.r. function f by a Σ_1 canonical wff which recapitulates a full definition for f.*

7.4 Q can capture all p.r. functions

We now want to show that not only can the language of Q express all p.r. functions, but can also capture them.

But hold on! We haven't yet said what it is for a theory to capture a function. So we first need to explain ...

7.4.1 How to capture a function

Recall our earlier definition Defn 16 which defines what it is to capture a property. The analogue for a two-place relations is of course

Defn. 35. *The theory T captures the two-place relation R by the open wff $\psi(\mathsf{x},\mathsf{y})$ iff, for any m, n,*

i. if m is R to n, then $T \vdash \psi(\overline{\mathsf{m}},\overline{\mathsf{n}})$,

ii. if m is not R to n, then $T \vdash \neg\psi(\overline{\mathsf{m}},\overline{\mathsf{n}})$

Thinking of $f(m) = n$ as stating a two-place relation between m and n we might expect the definition for capturing a function to have the same shape:

Defn. 36. *The theory T captures the one-place function f by the open wff $\chi(\mathsf{x},\mathsf{y})$ iff, for any m, n,*

i. if $f(m) = n$, then $T \vdash \chi(\overline{\mathsf{m}},\overline{\mathsf{n}})$,

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

But (for techie reasons we are not going to fuss about here) it turns out to be useful to add a further requirement

iii. $T \vdash \exists!\mathsf{y}\chi(\overline{\mathsf{m}},\mathsf{y})$.

In other words, T 'knows' that χ is functional (associates a number to just one value).

7.4.2 The proof strategy

We now want to show

Theorem 25. *The theory Q can capture any p.r. function by a Σ_1 wff.*

And there's more than one route to this theorem, including *a direct assault* and *a clever trick*. In the first edition of *IGT*, Ch. 13, I went for the clever trick. Which I now rather regret – for while clever tricks can give you a theorem they may not necessarily give you real understanding. So what I'll describe here is the direct assault which is pursued in the second edition. Basically, we just replicate the overall strategy for proving results about all p.r. functions which we described in §6.3, and deployed already in §6.4.1 and §7.2.

Suppose then that we can prove

C1. Q can capture the initial functions.

C2. If Q can capture the functions g and h, then it can also capture a function f defined by composition from g and h.

C3. If Q can capture the functions g and h, then it can also capture a function f defined by primitive recursion from g and h.

where in each case the capturing wffs are Σ_1. Then – by just the same sort of argument as in §7.2 – it follows that Q can capture any p.r. function by a Σ_1 wff.

So how do we prove C1? We just check that the formulae we said in §7.2 *express* the initial functions in fact serve to *capture* the initial functions in Q.

How do we prove C2? Suppose g and h are one-place functions, captured by the wffs $\mathsf{G}(\mathsf{x},\mathsf{y})$ and $\mathsf{H}(\mathsf{x},\mathsf{y})$ respectively. Then we prove that the function

$f(x) = h(g(x))$ is captured by the wff $\exists \mathsf{z}(\mathsf{G}(\mathsf{x}, \mathsf{z}) \wedge \mathsf{H}(\mathsf{z}, \mathsf{y}))$ (which is Σ_1 if G and H are).

And how do we prove C3? This is the tedious case that takes hard work, done in gory detail in second edition of the book! We need to show the formula B not only expresses but captures Gödel's β-function (and in fact we make use of a slightly stronger result). And then we use that fact to prove that if the n-place function g is captured by a $(n+1)$-variable expression G, and the $(n+2)$-variable function h by the $(n+3)$-variable expression H, then the rather horrid wff C* in §7.2 captures the function f defined by primitive recursion from g and h. Not surprisingly, details get messy (not difficult, just messy).

So the basic story is this. Take a full definition for defining a p.r. function, ultimately out of the initial functions. Follow the step-by-step instructions implicit in §7.2 about how to build up a canonical wff which in effect recapitulates that recipe. You'll get a wff which expresses the function, and that same wff captures the function in Q (and in any stronger theory with a language which includes L_A). Moreover the wff in question will be Σ_1.

7.5 Expressing/capturing properties and relations

Just a brief coda, linking what we've done in this episode with the last section of the previous episode.

We said in §6.6, Defn. 32 that the characteristic function c_P of a monadic numerical property P is defined by setting $c_P(m) = 0$ if m is P and $c_P(m) = 1$ otherwise. And a property P is said to be p.r. decidable if its characteristic function is p.r.

Now, suppose that P is p.r.; then c_P is a p.r. function. So, by Theorem 23, L_A can express c_P by a two-place open wff $\mathsf{c}_P(\mathsf{x}, \mathsf{y})$. So if m is P, i.e. $c_P(m) = 0$, then $\mathsf{c}_P(\overline{\mathsf{m}}, \mathsf{0})$ is true. And if m is not P, i.e. $c_P(m) \neq 0$, then $\mathsf{c}_P(\overline{\mathsf{m}}, \mathsf{0})$ is not true. So, by the definition of expressing-a-property, the wff $\mathsf{c}_P(\mathsf{x}, \mathsf{0})$ serves to express the p.r. property P. The point generalizes from monadic properties to many-place relations. So we have as an easy corollary of Theorem 23 that

Theorem 26. *L_A can express all p.r. decidable properties and relations.*

Similarly, suppose again that P is p.r. so c_P is a p.r. function. So, by Theorem 25, Q can capture c_P by a two-place open wff $\mathsf{c}_P(\mathsf{x}, \mathsf{y})$. So if m is P, i.e. $c_P(m) = 0$, so $\mathsf{Q} \vdash \mathsf{c}_P(\overline{\mathsf{m}}, \mathsf{0})$. And if m is not P, i.e. $c_P(m) \neq 0$, then $\mathsf{Q} \vdash \neg \mathsf{c}_P(\overline{\mathsf{m}}, \mathsf{0})$. So, by the definition of capturing-a-property, the wff $\mathsf{c}_P(\mathsf{x}, \mathsf{0})$ serves to capture the p.r. property P in Q. The point trivially generalizes from

monadic properties to many-place relations. So we have as an easy corollary of Theorem 25 that

Theorem 27. Q *can capture all p.r. decidable properties and relations.*

Further reading

Now read, if you must, *IGT2*, Chs 15–17. But quite honestly, you can live without knowing all the additional details there!

Episode 8

The arithmetization of syntax

- A very little on Hilbert's program
- Gödel coding
- The relation that holds between m and n when m codes for a PA proof of the wff with code n is p.r.
- The (standard) notation $\ulcorner\varphi\urcorner$ to denote to code number for the wff φ, and $\overline{\ulcorner\varphi\urcorner}$ as shorthand for the standard numeral for $\ulcorner\varphi\urcorner$.

This episode looks rather more substantial than it really is. The first page-and-a-bit reviews where we've been. And then there is some rough-and-ready historical scene-setting in a box which (depending on your background) you might well just want to skip through. The real action continues with §8.1 on p. 94.

Anyway, let's pause to take stock yet again. For more than half the battle is to get a clear sense of the overall route we are taking to proving Gödel's First Incompleteness Theorem. Keep that in sharp focus and you'll understand which are the Big Ideas that are in play and see too which bits of the proof are "boring details".

1. In Episode 1 we introduced the very idea of a formal axiomatized theory, the notion of negation incompleteness, and we stated (two versions of) Gödel's First Theorem.

2. Next, in Episode 2, we gave a proof of an incompleteness theorem that is weaker than Gödel's, since it doesn't tell us how to construct a true-but-unprovable sentence. But the proof is suggestive as it starts from the idea of a *sufficiently strong* theory, i.e. one that can 'capture' every decidable property P of numbers, in the sense (roughly) of proving case by case that n is P when it is, and proving that it isn't when it isn't.

3. The opening episodes, however, proceeded at a considerable level of abstraction. We talked in the first episode of a theory's 'containing a modest amount of arithmetic' without explaining how much that it is. Then in the second episode we talked of a theory's being 'sufficiently strong', promising that a 'sufficiently strong' theory is indeed modest, but again without saying how much arithmetic that involves. But in Episode 3 we started looking at some actual theories of arithmetic of various strengths. As a warm-up exercise we first looked at 'Baby Arithmetic' BA, a complete quantifier-free arithmetic, that 'knows' how to compute the results of additions and multiplications. Then we added quantifiers, replaced BA's axiom schemata with quantified axioms, added another axiom that says that every number other than zero is a successor, and we get Robinson Arithmetic Q. This is *boringly* incomplete – meaning that it can be shown to be incomplete without any fancy Gödelian arguments. It can't even prove $\forall \mathsf{x}(\mathsf{0} + \mathsf{x} = \mathsf{x})$. [Reality check: how is it that BA is complete and yet the stronger theory Q isn't?]

4. Q, as we said, is boringly incomplete for L_A truths. How can we beef it up at least so that it can prove elementary quantified truths like $\forall \mathsf{x}(\mathsf{0} + \mathsf{x} = \mathsf{x})$? By adding induction. If we add each instance of the so-called Induction Schema as an extra axiom to Q, we get First-Order Peano Arithmetic PA, described in Episode 4. This rich theory certainly proves all familiar elementary arithmetical claims than can be expressed in the first-order language L_A which has the successor, addition, and multiplication functions built in. In fact, pre-Gödel, the natural conjecture would be that it is a complete theory for the truths of L_A.

5. But of course, elementary arithmetic deals with many more functions than successor, addition, and multiplication, i.e. many more functions than are

built in to L_A. In Episode 6, we looked at a whole family of functions – the primitive recursive functions – which can defined in the sort of way that e.g. multiplication and addition are defined by recursion. These p.r. functions are computable functions, but aren't all of the computable functions.

6. Episode 7 showed that L_A can in fact *express* all the p.r. functions – i.e. for any one-place p.r. function f, we can construct in L_A a wff φ which is satisfied by a pair of numbers m, n just when $f(m) = n$ (and similarly for many-place functions). And we gestured towards a proof that even Q (and hence, a fortiori PA) can *capture* all the p.r. functions: if we construct the wff φ that expresses f in the right way, then if $f(m) = n$, $\mathsf{Q} \vdash \varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$, and if $f(m) \neq n$, $\mathsf{Q} \vdash \neg\varphi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$ (and Q knows φ is functional too). Moreover we can do the expressing/capturing job using wffs of low quantifier complexity, more precisely using Σ_1 wffs (these were defined back in the short Episode 5).

Now, that key result about what Q can capture doesn't *quite* take us back to the ideas in Episode 2. Earlier, we talked about 'sufficiently strong theories', where a theory is sufficiently strong if it captures all computably decidable properties of numbers. Episode 7 shows that Q and richer theories capture all p.r. decidable properties (meaning properties with p.r. characteristic functions). But since not all decidable properties are p.r. decidable (since not all computable functions are p.r. functions), this doesn't yet give us the result that Q and richer theories are sufficiently strong (even though they are).

However, it will turn out over this and the next episode that we don't need the stronger claim. Q's being able to be capture all p.r. functions is enough to make it the case that sensibly axiomatized theories that contain Q are incomplete (so long as they are consistent and satisfy another modest condition).

Now read on, perhaps pausing over a short historical interlude:

Formalization and finitary reasoning Think yourself back to the situation in mathematics a bit over a century ago. Classical analysis – the theory of differentiation and integration – has, supposedly, been put on firm foundations. We've done away with obscure talk about infinitesimals; and we've traded in an intuitive grasp of the continuum of real numbers for the idea of reals defined as 'Dedekind cuts' on the rationals or 'Cauchy sequences' of rationals. The key idea we've used in our constructions is the idea of a *set* of numbers. And we've been very free and easy with that, allowing ourselves

to talk of arbitrary sets of numbers, even when there is no statable rule for collecting the numbers into the set.

This freedom to allow ourselves to talk of arbitrarily constructed sets is just one aspect of the increasing freedom that mathematicians have allowed themselves, over the second half of the nineteenth century. We have loosed ourselves from the assumption that mathematics should be tied to the description of nature: as Morris Kline puts it, "after about 1850, the view that mathematics can introduce and deal with arbitrary concepts and theories that do not have any immediate physical interpretation ... gained acceptance". And Cantor could write "Mathematics is entirely free in its development and its concepts are restricted only by the necessity of being non-contradictory".

It is rather bad news, then, if all this play with freely created concepts, and in particular the fundamental notion of arbitrary sets, in fact gets us embroiled in contradiction – as seems to be the case as the set-theoretic paradoxes (like Russell's Paradox) pile up. What to do?

We might usefully distinguish two lines of responses that we can have to the paradoxes that threaten Cantor's paradise where mathematicians can play freely, which we might suggestively call the **foundationalist** and the **'more mathematics'** lines respectively

Foundationalist responses to paradox Consider first the option of seeking external "foundations" for mathematics.

We could, perhaps, seek to "re-ground" mathematics by confining ourselves again to applicable mathematics which has, as we would anachronistically put it, a model in the natural world so *must* be consistent.

The trouble with this idea is we're none too clear what this re-grounding in the world would involve – for remember, we are thinking back at the beginning of the twentieth century, as relativity and quantum mechanics are emerging, and any Newtonian confidence that we had about the real structure of the natural world is being shaken.

So maybe we have to put the option of anchoring mathematics in the physical world aside. But perhaps we could try to ensure that our mathematical constructions are grounded in the mental world, in mental constructions that we can perform and have a secure epistemic access to. This idea leads us to Brouwer's intuitionism. But it depends on an obscure notion of mental construction, and in any case – in its most worked out form – this approach

cripples lots of classical mathematics that we thought was unproblematically in good order, rather than giving it a foundation.

So what other foundational line might we take? How about trying to go back to incontrovertible logical principles and trying to find definitions of mathematical notions in logical terms, and seeking to constrain mathematics to what we can reconstruct on a firm *logical* footing.

This **logicist** line is perhaps more promising but it is still problematic. For remember, we are pretending to be situated a hundred or so years back, and at least at this point – leaving aside its beginnings in the as-yet-hardly-read work of Frege – modern logic itself isn't in as good a shape as most of the mathematics we are supposedly going to use it to ground (and indeed what might *count* as logic is still pretty obscure). Moreover, as Peirce saw, it looks as if we are going to need to appeal to mathematically developed ideas in order to develop logic itself; indeed Peirce himself thought that all formal logic is merely mathematics applied to logic. Still, perhaps we shouldn't give up yet. The proof is in the pudding: let's see if we can actually do the job and reconstruct mathematics on a logical basis and write a *Principia Mathematica* ...!

'Mathematical' responses to paradox Maybe, however, we just shouldn't be seeking to give mathematics a prior "foundation" after all. Consider: the paradoxes arise within mathematics, and to avoid them (the working mathematican might reasonably think) we just need to do the mathematics more carefully. As Peirce – for example – held, mathematics risks being radically distorted if we seek to make it answerable to some outside considerations. We don't need to look outside mathematics (to the world, to mental constructions, to logic) for a prior justification that will guarantee consistency. Rather *we need to improve our mathematical practice, in particular by improving the explicitness of our regimentations of mathematical arguments, to reveal the principles we actually use in 'ordinary' mathematics, and to see where the fatal mis-steps must be occurring when we over-stretch these principles in ways that lead to paradox.*

How do we improve explicitness, and pursue more careful explorations? A first step will be to work towards regimenting the principles that we actually need in mathematics into something approaching the ideal form of an effectively axiomatized formal theory. This is what Zermelo aimed to do in axiomatizing set theory: to locate the principles actually needed for

the seemingly 'safe' mathematical constructions needed in grounding classical analysis and other familiar mathematical practice. And when the job is done it seems that *these* principles don't in fact allow the familiar reasoning leading to Russell's Paradox or other set-theoretic paradoxes. So like the foundationalist/logicist camp, the 'do maths better' camp are also keen on battering a mathematical theory T into nice tidy axiomatized formats and sharply defining the rules of the game: but their purpose significantly different. The aim now is not to expose foundations, but to put our theory into a form that reveals it is as being indeed consistent and trouble-free.

For note this crucial insight of Hilbert's:

> The axiomatic formalization of a mathematical theory T (whether about widgets, wombats, or other whatnots), gives us *new* formal objects that are themselves apt topics for new formal mathematical investigations – namely the T-wffs and T-proofs that make up the theory!

And, crucially, when we go metatheoretical like this and move from thinking about *sets* (for example) to thinking about the syntactic properties of *formalized-theories-about-sets*, we move from considering suites of *infinite* objects (sets) to considering suites of *finite* formal objects (the wffs, and the finite sequences of wffs that form proofs). This means that we might then hope to bring to bear, at the metatheoretical level, entirely 'safe', merely *finitary*, reasoning about these suites of finite formal objects in order to prove consistency, etc.

Of course, it is a moot point what exactly constitutes such ultra-'safe' finitary reasoning. But still, it certainly looks as if we will – for instance – need much, much, less than full set theory to reason about formalized-set-theory as a suite of finite syntactic objects. So we might, in particular, hope with Hilbert – in the do-mathematics-better camp – to be able to use a safe uncontentious fragment of finitary mathematics to prove that our wildly infinitary set theory is at least syntactically consistent (doesn't prove both φ and $\neg\varphi$ for some wff φ).

So, at this point (I hope!) you can begin to see the attractions of what's called **Hilbert's Programme** – the programme of showing various systems of infinitary mathematics are contradiction-free by giving consistency proofs using safe, finitary, reasoning about the systems considered as formal objects.

Now, enter Gödel ...

8.1 Gödel-numbering

Hilbert's insight was that the syntactic objects that comprise formal theories (the wffs, the proofs) are *finite* objects, and so we only need mathematics of finite objects to theorize about the syntactic properties of theories.

But here comes Gödel's great twist on Hilbert's insight: *when we are dealing with finite objects, we can associate them with numerical codes: and then we can use* arithmetic *to talk about these codes and to deal with the arithmetical properties that – via the coding – 'track' syntactic properties of the theories.*

Nowadays, of course, the idea seems almost trite, for we are so very used to the notion that any finite data – in effect, any data that a finite computer can handle – can be digitized, i.e. rendered as binary numbers. But in the late 1920s this wonderfully simple but powerful idea must have seemed revelatory.

We'll implement it in stages. In this section, we do the basic coding.

8.1.1 Coding expressions in a formal language

We'll concentrate on the particular case of coding up expressions of the language L_A (but you'll see that the same basic idea, with modest variations, will work for any formal language). There are of course different ways of doing this: we'll follow in general style both Gödel and later tradition (though there *are* other codings on the market, which can smooth the path of some proofs).

So 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:

Defn. 37. *Suppose that the expression e is 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 $2^{c_0} \cdot 3^{c_1} \cdot 5^{c_2} \cdot \ldots \cdot \pi_k^{c_k}$.*

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

iii. The wff

$$\exists \mathsf{y}\,(\mathsf{S0} + \mathsf{y}) = \mathsf{SS0}$$

has the g.n.

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

That last number is, of course, *enormous*. So when we say that it is elementary to decode the resulting g.n. by taking the exponents of prime factors, we don't mean that the computation is quick. 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. And of course, the task produces a unique decoding by the fundamental theorem of arithmetic, that numbers are uniquely decomposable into prime factors.

8.1.2 Coding sequences

As well as talking about wffs via their code numbers, we'll want to talk about proofs via *their* code numbers. But how *do* we code for proof-arrays?

The details will obviously depend on the kind of proof system we adopt for the theory we are using. Suppose though, for simplicity, we consider theories with a Hilbert-style axiomatic system of logic. 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*.

Defn. 38. *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.

8.2 The arithmetization of syntactic properties/relatio

In this section, we'll continue to focus on the language L_A, and then on the particular theory PA which is built in that language. But again, similar remarks will apply mutatis mutandis to any sensibly built formal language, and any sensibly axiomatized formal theory.

8.2.1 *Term*, *Wff* and *Sent* are p.r. properties

We begin with some definitions (see §1.7):

Defn. 39. *Having fixed on our scheme of Gödel numbering, define the following numerical properties:*

1. *Term*(n) *is to hold when* n *codes for a term of* L_A.
2. *Wff*(n) *is to hold when* n *codes for a wff of* L_A.
3. *Sent*(n) *is to hold when* n *codes for a closed sentence of* L_A.

Then we have the following key result:

Theorem 28. *Term*(n), *Wff*(n), *and Sent*(n) *are p.r. decidable properties.*

Now, writing at the very beginning of the period when concepts of computation were being forged, Gödel 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, and so accepting Theorem 20, we can perhaps short-cut all that effort and be entirely persuaded by the following:

Proof. To determine whether *Term*(n), proceed as follows. Decode n: that's a mechanical exercise. Now ask: is the resulting expression 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? That's algorithmically decidable. The length of the first decoding stage of the computation will be bounded by a simple function of the length of n: similarly for the second stage of the computation, deciding whether the decoded expression – if there is one – is a term. Neither stage will involve any open-ended search, but rather can be done by programs using just bounded 'for' loops. Of course, these computations 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 using only 'for' loops operating on numbers, the numerical properties and relations which are decided by the whole procedure must be primitive recursive.

Similarly we can mechanically decide whether $Wff(n)$ or $Sent(n)$. Decode n again. Now ask: is the result an a wff or a sentence of L_A? In each case, that's algorithmically decidable, using programs involving nothing fancier than 'for' loops with bounds determined by the size of the input expression which n codes. And again, what's computably decidable using 'for' loops is primitive-recursively decidable. □

8.2.2 *Prf* is a p.r. relation

We introduce another definition:

Defn. 40. *Again having fixed on our scheme of Gödel numbering, the numerical relation $Prf(m, n)$ is defined to hold when m is the super g.n. of a proof in* PA *of the sentence with g.n. n.*

We have, as you might expect, a corresponding theorem:

Theorem 29. *$Prf(m, n)$ is a p.r. relation.*

Again we'll given an informal argument:

Proof. 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 next: is this sequence a properly constructed PA 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 can be done with 'for'-loops. □

A similar result, as we said, will hold for the corresponding Prf_T relation for any theory T which is like PA in respect of being axiomatized in such a way that we can mechanically check whether a string of wffs constitutes a T-proof using bounded 'for' loops. Any normally presented formalized theory will be like this. (In fact, we could reasonably have built that requirement into a sharper version of Defn. 3.)

Our results are robust! We've just shown that some properties and relations like *Term* and *Prf* are p.r. decidable. But note, $Term(n)$ – for example– is to hold when n is the code number of a term of L_A *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 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 involve at most bounded 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 (strictly, we need to build in a way of handling the 'waste' cases where n_1 isn't an S-code for any wff). 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:

Defn. 41. *A coding scheme S for L_A mapping expressions to numbers 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.

We immediately have

Theorem 30. *A property like Term defined using our official Gödelian coding 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.*

Proof. Let the characteristic functions of *Term* 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). □

In sum, our result that *Term* is p.r. is robust with respect to any sensible choice of coding scheme: similarly with the other results, in particular the result about *Prf*.

8.3 Some cute notation

Finally in this episode, we introduce a really pretty bit of notation.

Assume we have chosen some system for Gödel-numbering the expressions of some language L. Then

Defn. 42. *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. (Sometimes, we'll write the likes of $\ulcorner\mathsf{U}\urcorner$ where U abbreviates an L_A wff: we mean here, of course, the Gödel-number for the unabbreviated original wff that U stands in for.)

So far so good. But in the book, I use this very same notation also 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 in the book, 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. I could have alternatively used in the book the common practice of always overlining abbreviations for standard numerals: we would then indicate the numeral for the g.n. number $\ulcorner\mathsf{SS0}\urcorner$ by the slightly messy '$\overline{\ulcorner\mathsf{SS0}\urcorner}$'. Many writers do this. But I thought that aesthetics recommended my fairly common and rather prettier convention.

I still think that was a reasonable decision: but in these notes, in the interests of maximal clarity – if only as a helpful ladder that you can throw away once climbed! – I *will* here use the clumsier notation. So, to avoid any possibly misunderstandings, we'll adopt:

Defn. 43. *Used in L-expressions, '$\overline{\ulcorner\varphi\urcorner}$' is shorthand for L's standard numeral for the g.n. of φ.*

So: naked corner quotes belong to augmented English; overlined corner quotes are an abbreviatory device in the relevant formal language L.

Further reading

IGT2, Chs. 19 and 20.

Episode 9

The First Incompleteness Theorem

- The idea of diagonalization
- How to construct a 'canonical' Gödel sentence
- If PA is sound, it is negation incomplete
- Generalizing that result to sound p.r. axiomatized theories whose language extends L_A
- ω-incompleteness, ω-inconsistency
- If PA is ω-consistent, it is negation incomplete
- Generalizing that result to ω-consistent p.r. axiomatized theories which extend Q
- The historical First Theorem

The pieces we need to prove the First Theorem are finally all in place. So in this episode we at long last learn how to construct 'Gödel sentences' and use them to prove that PA is incomplete. We also show how to generalize the result to other theories.

Let's quickly review (not the whole route to where we have got but) the key ingredients that need to be in place for the arguments to come. You need to understand the following:

i. We can fix on some acceptable scheme for coding up wffs of PA's language L_A by using Gödel numbers ('g.n.' for short), and coding up PA-proofs – i.e. sequences or other arrays of wffs – by super Gödel numbers. Similarly later for coding up wffs and proofs of other theories. (§8.2)

ii. Notation: If φ is an expression, then we'll denote its Gödel number in our logician's English by '$\ulcorner\varphi\urcorner$'. We use '$\overline{\ulcorner\varphi\urcorner}$' as an abbreviation inside L_A for the standard numeral for $\ulcorner\varphi\urcorner$. Note that later, when we start generalizing Gödel's results to other theories, we'll use the same notation for Gödel numberings of other languages. (§8.3)

iii. $Prf(m, n)$ is the 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 (assume we've fixed on some definite version of PA). This relation is p.r. decidable. Similarly for other theories. (§8.2.2)

iv. Any p.r. function or relation can be *expressed* by a wff of PA's language L_A. In particular, we can choose a Σ_1 wff which '*canonically*' expresses a given p.r. relation by recapitulating its p.r. definition (or more strictly, by recapitulating the definition of the relation's characteristic function). (§7.2)

v. Any p.r. function or relation can be *captured* in Q and hence in PA (and captured by a Σ_1 wff which canonically expresses it). (§7.4)

For what follows, it isn't necessary that you know the *proofs* of the claims we've just summarized: but do check that you at least fully understand what the various claims *say*.

9.1 The idea of diagonalization

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 (assuming PA is consistent). We now have an inkling of how he can do that: wffs can contain numerals which refer to numbers which – via Gödel coding – are in turn correlated with wffs. Maybe we can, if we are

cunning, get a wff to be – via the coding – about itself. And the next section shows how.

But first we need to introduce a simple but pivotal construction. We will say

Defn. 44. *The diagonalization of a wff φ with one free variable is the wff $\varphi(\overline{\ulcorner\varphi\urcorner})$.*

That is to say, the diagonalization of the wff φ is what you get by substituting for its free variable the numeral for the Gödel number of φ itself. (We can do this construction in any suitable language, of course, but for the moment let's concentrate on L_A.)

Why is this substitution operation called *diagonalization*? Well compare the 'diagonal' construction we encountered in §2.3, where we counted off wffs $\varphi_0, \varphi_1, \varphi_2 \ldots$ in an enumeration of wffs with one free variable, and substituted (the numeral for) the index n for the free variable in the wff φ_n, to form $\varphi_n(\overline{\mathsf{n}})$. Here, in our Gödelian diagonal construction, we take the Gödel number of a wff with one free variable – and we can think of the Gödel number as indexing the wff in a list of wffs – and substitute the numeral for this for the free variable.

Diagonalization defined like this (or in the slightly variant way that I use in the book) is, evidently, a very simple mechanical operation on expressions. In fact,

Theorem 31. *There is a p.r. function $diag(n)$ which, when applied to a number n which is the g.n. of some L_A wff with one free variable, yields the g.n. of that wff's diagonalization, and yields n otherwise.*

Proof. Consider this procedure. Try treating n as a g.n., and seek to decode it. If you don't get an expression with one free variable, return n. Otherwise you get a wff φ and can form the wff $\varphi(\overline{\mathsf{n}})$, which is its diagonalization. Then work out the g.n. of this result to compute $diag(n)$. This procedure doesn't involve any unbounded searches. So we again will be able to program the procedure using just 'for' loops. Hence *diag* is a p.r. function □

And since Q captures every p.r. function, that means in particular we have

Theorem 32. *There is wff $\mathsf{Diag}(\mathsf{x},\mathsf{y})$ which (canonically) captures diag in Q.*

9.2 Constructing a Gödel sentence

In this section, we construct a Gödel sentence for PA in particular. But the mode of construction will evidently generalize – a point we return to in the next section.

First, another definition:

Defn. 45. *The relation* $Gdl(m, n)$ *is defined to hold just when* m *is the super g.n. for a* PA *proof of the diagonalization of the wff with g.n.* n.

Theorem 33. $Gdl(m, n)$ *is p.r. decidable.*

Proof. We just remark that, as with $Prf(m, n)$, we can mechanically check whether $Gdl(m, n)$ using simple 'for'-loops (and no open-ended searches). □

> You want a fancier argument? Well, we can note that $Gdl(m, n)$ holds, by definition, when $Prf(m, diag(n))$. Let c_{Prf} be the characteristic function of Prf, which is p.r.: then $c_{Gld}(x, y) =_{\text{def}} c_{Prf}(x, diag(y))$ will be the characteristic function of Gld, so – being a composition of p.r. functions – c_{Gld} is p.r. too.

Now Gdl can be expressed in L_A by a Σ_1 wff, which in fact captures Gdl in PA (indeed, in Q), and does so by recapitulating a full definition of the (characteristic function) of the p.r. relation. So let's say

Defn. 46. $\mathsf{Gdl}(\mathsf{x}, \mathsf{y})$ *stands in for some* Σ_1 *wff which canonically expresses and captures* Gdl.

And we next follow Gödel in first constructing the corresponding wff

Defn. 47. $\mathsf{U}(\mathsf{y}) =_{\text{def}} \forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \mathsf{y})$.

Now we diagonalize U, to give

Defn. 48. $\mathsf{G} =_{\text{def}} \mathsf{U}(\overline{\ulcorner \mathsf{U} \urcorner}) = \forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \overline{\ulcorner \mathsf{U} \urcorner})$.

And here is the wonderful result:

Theorem 34. G *is true if and only if it is unprovable in* PA.

Proof. Consider what it takes for G to be true (on the interpretation built into L_A of course), given that the formal predicate Gdl expresses the numerical relation Gdl.

G is true if and only if for all numbers m it isn't the case 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. □

G – meaning of course the L_A sentence you get when you unpack the abbreviations! – is thus our 'Gödel sentence' for PA. We might indeed call it a *canonical* Gödel sentence for three reasons: (a) it is defined in terms of a wff that we said canonically expresses/captures *Gdl*, and (b) because it is roughly the sort of sentence that Gödel himself constructed, so (c) it is the kind of sentence people standardly have in mind when they talk of '*the*' Gödel sentence for PA.

Note that G will be horribly long when spelt out in unabbreviated L_A. But in another way, it is relatively simple. We have the easy result that

Theorem 35. G *is* Π_1.

Proof. $\mathsf{Gdl}(\mathsf{x}, \mathsf{y})$ is Σ_1. So $\mathsf{Gdl}(\mathsf{x}, \overline{\ulcorner \mathsf{U} \urcorner})$ is Σ_1. So its negation $\neg\mathsf{Gdl}(\mathsf{x}, \overline{\ulcorner \mathsf{U} \urcorner})$ is Π_1. Hence $\forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \overline{\ulcorner \mathsf{U} \urcorner})$ is Π_1 too. □

What G says 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. However, such a claim is never *strictly* true; though if we do restrict ourselves to *canonical* Gödel sentences, then we might reasonably claim they *indirectly* say that they are unprovable, or some such.

First, let's stress that any that G (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 '$\times$', which all have the standard interpretation built into L_A. In particular, the standard numeral in G refers to a number, not a wff.

However there is perhaps a reasonable sense in which G *can* be described as signifying or *indirectly* saying that it is unprovable. 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*. 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 a 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 signifies that it is unprovable.

9.3 The First Theorem – the semantic version

9.3.1 If PA is sound, it is incomplete

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, then PA *would* prove a false theorem, contradicting our supposition. Hence, G is not provable in PA.

But that shows that G *is* true. So ¬G must be false. Hence ¬G cannot be proved in PA either, supposing PA is sound. In Gödel's words, G is a 'formally undecidable' sentence of PA (see Defn. 7).

Which establishes

Theorem 36. *If* PA *is sound, then there is a true* Π_1 *sentence* G *such that* $\mathsf{PA} \nvdash \mathsf{G}$ *and* $\mathsf{PA} \nvdash \neg\mathsf{G}$*, so* PA *is negation incomplete.*

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 – or at least, it's that simple once we have constructed G.

9.3.2 Generalizing the proof

The proof evidently generalizes. Suppose T is any theory at all, that is put together so that we can mechanically check whether a purported T-proof is indeed a kosher proof in a nicely bounded proof that can be regimented using nothing more exotic then 'for' loops. Then, assuming a sensible scheme for Gödel-number

wffs of T, the relation $Prf_T(m, n)$ which holds when m numbers a proof of the wff with number n will be primitive recursive again. Let's say that a theory is **p.r. axiomatized** when it is indeed axiomatized so as to make Prf_T primitive recursive: then indeed any normal theory you dream up which is formally axiomatized is p.r. axiomatized.

Suppose now that T's language includes the language of basic arithmetic, L_A (see Defn. 9), so T can form standard numerals, and we can form the diagonalization of a T-wff. Then we can also define the relation $Gld_T(m, n)$ which holds when m numbers a T-proof of the diagonalization of the wff with number n. This too will be primitive recursive again.

Continuing to suppose that T's language includes the language of basic arithmetic, T will be able to express the p.r. relation Gld_T by a Σ_1 wff Gld_T. Then, just as we did for PA, we'll be able to construct the corresponding Π_1 wff G_T. And then exactly the same argument as before will show, more generally,

Theorem 37. *If T is a sound p.r. axiomatized theory whose language contains the language of basic arithmetic, then there will be a true Π_1 sentence G_T such that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$, so T is negation incomplete.*

Which is our first, '**semantic**', version of the **First Incompleteness Theorem**!

Let's note an immediate corollary:

Theorem 38. *There is no p.r. axiomatized theory framed in the language of L_A whose theorems are all and only the truths of L_A.*

9.3.3 Our Incompleteness Theorem is better called an *incompletability* theorem

Here, we just repeat the argument of §1.6: but the point is central enough to bear repetition. Suppose T is a sound p.r. axiomatized theory which can express claims of basic arithmetic. Then by Theorem 37 we can find a true G_T such that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$. That *doesn't* mean that G_T is 'absolutely unprovable' in any sense: it just means that G_T-is-unprovable-in-T.

Now, we might want to 'repair the gap' in T by adding G_T as a new axiom. So consider the theory $T' = T + \mathsf{G}_T$. Then (i) T' is still sound (for the old T-axioms are true, the added new axiom is true, and the logic is still truth-preserving). (ii) T is evidently still a p.r. axiomatized theory (why?). (iii) We haven't changed the language. So our Incompleteness Theorem applies, and we can find a sentence G'_T such that $T' \nvdash \mathsf{G}_{T'}$ and $T' \nvdash \neg\mathsf{G}_{T'}$. And since T' is stronger than T, we have a fortiori, $T \nvdash \mathsf{G}_{T'}$ and $T \nvdash \neg\mathsf{G}_{T'}$. In other words, 'repairing the gap' in T by

adding G_T as a new axiom leaves some other sentences that are undecidable in T *still* undecidable in the augmented theory.

And so it goes. Our theorem tells us that if we keep chucking more and more additional axioms at T, our theory will still remain negation-incomplete, unless it either stops being sound or stops being p.r. axiomatized. In a good sense, T is *incompletable*.

Comparing old and new semantic incompleteness theorems Compare our new Theorem 37 with our initially announced

Theorem 1. *If T is a sound formal axiomatized theory whose language contains the language of basic arithmetic, then there will be a true sentence G_T of basic arithmetic such that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$, so T is negation incomplete.*

Our new theorem is stronger in one respect, weaker in another. But the gain is much more than the loss.

Our new theorem is stronger, because it tells us more about the character of the undecidable Gödel sentence – namely it has minimal quantifier complexity. The unprovable sentence G_T is a Π_1 sentence of arithmetic, i.e. is the universal quantification of a decidable condition. As far as quantifier complexity is concerned, it is on a par with e.g. Goldbach's conjecture that every number is such that, if even and greater than two primes, it is the sum of two (for note it is decidable whether a number is the sum of two primes). Indeed it is sometimes said that a Gödel sentence like G_T is **of Goldbach type**.

Our new theorem is weaker, however, as it only applies to p.r. axiomatized theories, not to effectively axiomatized theories more generally. But that's not much loss. For what would a theory look like that was effectively axiomatized but not p.r. axiomatized? It would be a matter, for example, of only being able to tell what's an axiom on the basis of an open-ended search: but that would require a *very* unnatural way of specifying the theorem's axioms in the first place. As we just noted, any normally presented effectively axiomatized theory will be p.r. axiomatized.

Now, we can in fact extend Gödel's theorem to cover the case of abnormally though still effectively decidably axiomatized theories. In a way, however, that really is a minor extension. And to do all the work of getting

clear about the broader notion of the effectively computable – as opposed to the primitive recursive – takes quite a bit of work, which we undertake in the later chapters of *IGT2*. Interesting and important though all that is, you don't really need to get your head round all that to get a basic grasp on Gödelian incompleteness.)

Now do pause here: have a think, have a coffee!

Are you absolutely clear about how G is constructed? Are you absolutely clear why it is true if and only if unprovable? Do you understand why it it must be formally undecidable assuming PA is sound? Do you understand how and why the result generalizes?

If you answer 'no' to any of those, re-read more carefully! If you answer 'yes' to all, excellent: on we go ...

9.4 ω-completeness, ω-consistency

Before we turn to the second version of the First Incompleteness Theorem – the version that downgrades the semantic assumption that we're dealing with a sound theory to the much weaker syntactic assumption that the theory is consistent (and a bit more) – we need to pause to define two key notions.

Techie note: in this section, take the quantifiers mentioned to be arithmetical ones – if necessary, therefore, replacing $\forall \mathsf{x}\varphi(\mathsf{x})$ by $\forall \mathsf{x}(\mathsf{N}\mathsf{x} \to \varphi(\mathsf{x}))$, where '$\mathsf{N}$' picks out the numbers from the domain of the theory's native quantifiers (see Defn. 9).

Defn. 49. *A theory T is* ω-incomplete *iff, for some open wff $\varphi(\mathsf{x})$, T can prove $\varphi(\overline{\mathsf{n}})$ for each natural number n, but T can't go on to prove $\forall \mathsf{x}\varphi(\mathsf{x})$.*

We saw in §3.5 that Q is ω-incomplete: that's because it can prove each instance of $\mathsf{0} + \overline{\mathsf{n}} = \overline{\mathsf{n}}$, but can't prove $\forall \mathsf{x}(\mathsf{0} + \mathsf{x} = \mathsf{x})$. We could repair ω-incompleteness if we could add the ω-rule (see §4.1.1), but that's an infinitary rule that is not available in a formalized theory given the usual finitary restrictions on the checkability of proofs. We instead added induction to Q hoping to repair as much incompleteness as we could: but, as we'll see, PA remains ω-incomplete (assuming it is consistent).

Defn. 50. *A theory T is* ω-inconsistent *iff, for some open wff $\varphi(\mathsf{x})$, T can prove each $\varphi(\overline{\mathsf{n}})$ and T can also prove $\neg\forall \mathsf{x}\varphi(\mathsf{x})$.*

Or, entirely equivalently, we could of course say that T is ω-inconsistent if, for some open wff $\varphi'(\mathsf{x})$, $T \vdash \exists \mathsf{x}\varphi'(\mathsf{x})$, yet for each number n we have $T \vdash \neg\varphi'(\overline{\mathsf{n}})$.

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, in a classical context, ω-consistency – defined of course as not being ω-inconsistent! – trivially 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 compare and contrast. Suppose T can prove $\varphi(\overline{\mathsf{n}})$ for each n. 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 $\neg\forall \mathsf{x}\varphi(\mathsf{x})$.

So 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 $\varphi(\overline{\mathsf{n}})$ and yet also prove $\neg\forall \mathsf{x}\varphi(\mathsf{x})$ is just not going to be an acceptable candidate for regimenting arithmetic.

That last observation can be made vivid if we temporarily bring semantic ideas back into play. Suppose the theory T is given a *arithmetically standard* interpretation, by which we here mean just an interpretation which takes numerical quantifiers as running over a domain comprising 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 $\varphi(\mathsf{0})$, $\varphi(\overline{\mathsf{1}})$, $\varphi(\overline{\mathsf{2}})$, ... 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 39. *If T is ω-inconsistent then T's axioms can't all be true on an arithmetically standard 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 e.g. PA *is* sound on its standard interpretation, we are committed to thinking that it *is* ω-consistent.

9.5 The First Theorem – the syntactic version

9.5.1 If PA is consistent, it can't prove G

So far, we have actually only made use of the weak result that PA's language can *express* the relation *Gdl*. But remember Defn. 46: our chosen Gdl doesn't just express *Gdl* but *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.

Theorem 40. *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) $\mathsf{PA} \vdash \mathsf{Gdl}(\overline{\mathsf{m}}, \overline{\ulcorner \mathsf{U} \urcorner})$.

But since G is none other than $\forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \overline{\ulcorner \mathsf{U} \urcorner})$, the assumption that G is provable comes to this: $\mathsf{PA} \vdash \forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \overline{\ulcorner \mathsf{U} \urcorner})$. The universal quantification here entails any instance. Hence (ii) $\mathsf{PA} \vdash \neg \mathsf{Gdl}(\overline{\mathsf{m}}, \overline{\ulcorner \mathsf{U} \urcorner})$.

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. □

9.5.2 If PA is consistent, it is ω-incomplete

Here's an immediate corollary of that last theorem:

Theorem 41. *If* PA *is consistent, it is* ω*-incomplete.*

Proof. Assume PA's consistency. Then we've shown that $\mathsf{PA} \nvdash \mathsf{G}$, i.e.,

1. $\mathsf{PA} \nvdash \forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \overline{\ulcorner \mathsf{U} \urcorner})$.

Since G is unprovable, that means that 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, $\mathsf{PA} \vdash \neg \mathsf{Gdl}(\overline{\mathsf{m}}, \overline{\ulcorner \mathsf{U} \urcorner})$.

Putting $\varphi(\mathsf{x}) =_{\text{def}} \neg \mathsf{Gdl}(\mathsf{x}, \overline{\ulcorner \mathsf{U} \urcorner})$, the combination of (1) and (2) therefore shows that PA is ω-incomplete. □

9.5.3 If PA is ω-consistent, it can't prove $\neg$G

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.

Theorem 42. *If* PA *is ω-consistent,* $\mathsf{PA} \nvdash \neg\mathsf{G}$.

Proof. Suppose $\neg$G is provable in PA. That's equivalent to assuming

1. $\mathsf{PA} \vdash \exists x \mathsf{Gdl}(x, \overline{\ulcorner \mathsf{U} \urcorner})$.

Now suppose too that PA is ω-consistent. Then, as we remarked before, that implies that PA 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

2. $\mathsf{PA} \vdash \neg\mathsf{Gdl}(\overline{\mathsf{m}}, \ulcorner \mathsf{U} \urcorner)$ for each m.

But (1) and (2) together make PA ω-inconsistent after all, contrary to hypothesis. Hence, if PA is ω-consistent, $\neg$G is unprovable. □

9.5.4 Putting together the syntactic Incompleteness Theorem for PA

Let's put all the ingredients together. Recall that G is a Π_1 sentence (i.e. of the same quantifier complexity as e.g. Goldbach's Conjecture). That observation put together with what we've shown so far this section entails

Theorem 43. *If* PA *is consistent, then there is a* Π_1 *sentence* G *such that* $\mathsf{PA} \nvdash \mathsf{G}$*, and if* PA *is ω-consistent* $\mathsf{PA} \nvdash \neg\mathsf{G}$*, so – assuming ω-consistency and hence consistency –* PA *is negation incomplete.*

9.5.5 Generalizing the proof

The proof for Theorem 43 evidently generalizes. Suppose T is a p.r. axiomatized theory which contains Q – so (perhaps after introducing some new vocabulary by definitions) the language of T extends the language of basic arithmetic, and T can prove Q's axioms. Then assuming a sensible scheme for Gödel-numbering wffs of T, the relation $Gdl_T(m, n)$ which holds when m numbers a T-proof of the diagonalization of the wff with number n will be primitive recursive again.

Since T can prove everything Q proves, T will be able to capture the p.r. relation Gld_T by a Σ_1 wff Gld_T. Just as did for PA, we'll be able to construct the corresponding Π_1 wff G_T. And, exactly the same arguments as before will then show, more generally,

Theorem 44. *If T is a consistent p.r. axiomatized theory which contains Q, then there will be a Π_1 sentence G_T such that $T \nvdash \mathsf{G}_T$, and if T is ω-consistent, $T \nvdash \neg\mathsf{G}_T$, so T is negation incomplete.*

When people refer to the **First Incompleteness Theorem** (without qualification), they typically mean something like this second general result, deriving incompleteness from **syntactic** assumptions.

Let's emphasize that last point. Being p.r. axiomatized is a syntactic property; containing Q is a matter of Q's axioms being derivable; being consistent here is a matter of no contradictory pair φ, $\neg\varphi$ being derivable – all syntactic notions. The chains of argument that lead to this theorem depend just on the given syntactic assumptions, via e.g. the proof that Q can capture all p.r. functions – another claim about a syntactically definable property. That's why I'm calling this the syntactic Incompleteness Theorem. (Of course, we are *interested* in various syntactically definable properties because of their semantic relevance: for example, we care about the idea of capturing p.r. functions because we are interested in what an interpreted theory might be able to prove in the sense of establish-as-true. But it is one thing for us to have a semantic motivation for being interested in a certain concept, it is another thing for that concept to have semantic content.)

Comparing old and new syntactic incompleteness theorems Compare Theorem 44 with our initially announced

Theorem 2. *For any consistent formal axiomatized theory T which can prove a certain modest amount of arithmetic (and has a certain additional desirable syntactically definable property that any sensible formalized arithmetic will share), there is a sentence of basic arithmetic G_T such that $T \nvdash \mathsf{G}_T$ and $T \nvdash \neg\mathsf{G}_T$, so T is negation incomplete.*

Our new theorem fills out the old one in various respects, but it is weaker in another respect. But the gain is much more than the loss.

Our new theorem tells us more about the 'modest amount of arithmetic' that T is assumed to contain and it also spells out the 'additional desirable

property' which we previously left mysterious (and we now know the condition is only applied in half the theorem). Further it tells us more about the undecidable Gödel sentence – namely it has minimal quantifier complexity, i.e. it is a Π_1 sentence of arithmetic. Our new theorem is weaker, however, as it only applies to p.r. axiomatized theories, not to formal axiomatized theories more generally. But we've already noted in the last box that that's not much loss. (And we can in fact go on to make up the shortfall).

9.6 Gödel's own Theorem

As we said, Theorem 44, or something like it, is what people usually mean when they speak without qualification of 'The First Incompleteness Theorem'. But since the stated theorem refers to Robinson Arithmetic Q (developed by Robinson in 1950!), and Gödel didn't originally know about that (in 1931), our version can't be quite what Gödel originally proved. 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 theories which can capture p.r. relations like Gdl*. It's being able to capture *Gdl* that is the crucial condition for a theory's being incomplete. So let's say

Defn. 51. *A theory T is* p.r. adequate *if it can capture all primitive recursive functions and relations.*

Then, instead of mentioning Q, let's instead explicitly write in the requirement of p.r. adequacy. So, by just the same arguments,

Theorem 45. *If T is a p.r. adequate, p.r. axiomatized theory whose language includes L_A, then there is Π_1 sentence φ such that, if T is consistent then $T \nvdash \varphi$, and if T is ω-consistent then $T \nvdash \neg\varphi$.*

And this is pretty much Gödel's own general version of the incompleteness result. I suppose that it has as much historical right as any to be called *Gödel's* First Theorem. ('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 does actually apply to familiar arithmetics, does depend on proving the adequacy theorem.)

Thus, in his 1931 paper, Gödel first proves his Theorem VI, which with a bit of help from his Theorem VIII shows that the formal system P – which is his simplified version of the hierarchical type-theory of *Principia Mathematica* – has a formally undecidable Π_1 sentence (or sentence 'of Goldbach type', see the box on p. 108). 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.

Which gives us our Theorem 45.

Further reading

At this point, make sure you really understand at least what the core theorems in this episode *mean*. Then read *IGT2*, Chs. 21 and 22.

And then re-read those chapters! – for they are at the very heart of the book.

Then when you feel reasonably confident of the techie details, have a look at Ch. 23.

Episode 10

The Diagonalization Lemma, Rosser and Tarski

- The Diagonalization Lemma
- Provability predicates
- Incompleteness from the Diagonalization Lemma
- Tarski's Theorem
- The Master Argument
- Rosser's Theorem

We've now proved our key version of the First Theorem, Theorem 44. Here's how we got there:

1. Our opening two Episodes were motivational. In Episodes 3 and 4 we got down to business, looking at some actual theories of arithmetic of various strengths. First we first looked at 'Baby Arithmetic' BA, a complete quantifier-free arithmetic, that 'knows' how to compute the results of additions and multiplications. Then we added quantifiers, replaced BA's axiom schemata with quantified axioms, added another axiom that says that every number other than zero is a successor, and we got Robinson Arithmetic

Q. This is *boringly* incomplete – meaning that it can be shown to be incomplete without any fancy Gödelian arguments. Then we added (all instances of) the so-called Induction Schema as an extra axiom to get First-Order Peano Arithmetic PA. Pre-Gödel, the natural conjecture would be have been that it is a complete theory for the truths of L_A.

2. Elementary informal arithmetic deals with many more functions than successor, addition, and multiplication. In Episode 6, we looked at a very large family of computable numerical functions, namely the p.r. functions. Episode 7 then showed that L_A can in fact *express* all the p.r. functions, and we gestured towards a proof that even Q (and hence, a fortiori PA) can *capture* all the p.r. functions. We can do the expressing/capturing job using wffs of low quantifier complexity, more precisely using Σ_1 wffs (these were defined back in the short Episode 5).

3. Episode 8 introduced the project of arithmetizing syntax. We explained the idea of Gödel numbering, coding expressions and sequences of expressions by numbers. And then we defined various numerical properties and relations such as $Prf(m, n)$ which holds when m is the super Gödel-number of a sequence of wffs constituting a PA-proof of a sentence with Gödel-number n. We remarked that these various numerical properties and relations are all p.r. (a result which isn't sensitive to the precise choice of Gödel-numbering system).

4. We need one more ingredient before constructing a Gödel sentence and proving incompleteness, the idea of diagonalization, introduced at the beginning of Episode 9. The diagonalization of a wff φ with one free variable is the wff $\varphi(\overline{\ulcorner\varphi\urcorner})$.

5. In the next part of Episode 9, we first defined the p.r. 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. Now, PA can express and capture this relation with a wff $\mathsf{Gdl}(\mathsf{x}, \mathsf{y})$. So form the wff $\mathsf{U}(\mathsf{y}) =_{\text{def}} \forall\mathsf{x}\neg\mathsf{Gdl}(\mathsf{x}, \mathsf{y})$, and let G be its diagonalization. Using just the fact that Gdl expresses Gdl, it is easy to see that G is true if and only if it is not PA-provable. It is immediate that if PA is sound, then PA cannot prove either G or $\neg$G. That's the semantic version of the First Incompleteness Theorem for PA. The result then easily generalizes to other sound p.r axiomatized theories whose language is at least as rich as L_A.

6. In the rest of Episode 9, we showed that, given the weaker assumption that PA is consistent, we can again show that PA cannot prove G (*now* we have to use the fact that Gdl captures *Gdl*). And if we make the stronger syntactic assumption that PA is ω-consistent, then PA cannot prove $\neg$G either. That's the syntactic version of the First Incompleteness Theorem for PA. This result too then easily generalizes to other p.r axiomatized theories which include Q.

Now moving on, this episode starts by proving the key syntactic incompleteness theorem again by a slightly different route – via the so-called Diagonalization Lemma. The interest in doing this is that the same Lemma leads to two other important theorems due to Rosser and Tarski.

10.1 The Diagonalization Lemma

First some reminders of previous definitions:

Defn.. 36. *The theory T captures the one-place function f by the open wff $\chi(\mathsf{x},\mathsf{y})$ iff, for any m, n,*

i. if $f(m) = n$, then $T \vdash \chi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$,
ii. if $f(m) \neq n$, then $T \vdash \neg\chi(\overline{\mathsf{m}}, \overline{\mathsf{n}})$.
iii. $T \vdash \exists!\mathsf{y}\chi(\overline{\mathsf{m}}, \mathsf{y})$.

Note that (i) and (iii) together imply (iv) if $f(m) = n$, then $T \vdash \forall\mathsf{x}(\chi(\overline{\mathsf{m}}, \mathsf{z}) \leftrightarrow \mathsf{z} = \overline{\mathsf{n}})$.

Defn. 44. *The diagonalization of a wff φ with one free variable is the wff $\varphi(\overline{\ulcorner\varphi\urcorner})$.*

Theorem 31. *There is a p.r. function diag(n) which, when applied to a number n which is the g.n. of some L_A wff with one free variable, yields the g.n. of that wff's diagonalization, and yields n otherwise.*

And generalizing Theorem 32, we have

Theorem 46. *There is a T-wff $\mathsf{Diag}_T(\mathsf{x},\mathsf{y})$ which captures diag in* Q.

And now we can officially state and then prove the following so-called **Diagonalization Lemma**:

Theorem 47. *If T extends* Q*, and φ is a one-place open sentence of T's language, then there is sentence δ such that (i) $\delta \leftrightarrow \varphi(\overline{\ulcorner\delta\urcorner})$ is true, and moreover (ii) $T \vdash \delta \leftrightarrow \varphi(\overline{\ulcorner\delta\urcorner})$.*

(Carnap noted the result (i): he is often, but wrongly, attributed (ii) as well.)

To avoid unsightly rashes of subscripts, let's henceforth drop subscript Ts. Then we can argue like this:

Proof for (i). Put $\alpha =_{\text{def}} \forall \mathsf{z}(\mathsf{Diag}(\mathsf{y}, \mathsf{z}) \rightarrow \varphi(\mathsf{z}))$, and let δ be the diagonalization of α. So δ is $\forall \mathsf{z}(\mathsf{Diag}(\overline{\ulcorner \alpha \urcorner}, \mathsf{z}) \rightarrow \varphi(\mathsf{z}))$.

Since diagonalizing α yields δ, we have $diag(\ulcorner \alpha \urcorner) = \ulcorner \delta \urcorner$. So since Diag expresses *diag*, we know that $\mathsf{Diag}(\overline{\ulcorner \alpha \urcorner}, \mathsf{z})$ is satisfied by $\ulcorner \delta \urcorner$ and no other number. So δ is true and only if $\ulcorner \delta \urcorner$ also satisfies $\varphi(\mathsf{z})$. Hence $\delta \leftrightarrow \varphi(\overline{\ulcorner \delta \urcorner})$ is true. □

Proof for (ii). Since by hypothesis Diag captures *diag* in Q and hence in T, we can use (iv) above to get $T \vdash \forall \mathsf{z}(\mathsf{Diag}(\overline{\ulcorner \alpha \urcorner}, \mathsf{z}) \leftrightarrow \mathsf{z} = \overline{\ulcorner \delta \urcorner})$.

But by definition, δ just is $\forall \mathsf{z}(\mathsf{Diag}(\overline{\ulcorner \alpha \urcorner}, \mathsf{z}) \rightarrow \varphi(\mathsf{z}))$. So using the proven equivalence of $\mathsf{Diag}(\overline{\ulcorner \alpha \urcorner}, \mathsf{z})$ and $\mathsf{z} = \overline{\ulcorner \delta \urcorner}$ we can derive $T \vdash \delta \leftrightarrow \forall \mathsf{z}(\mathsf{z} = \overline{\ulcorner \delta \urcorner} \rightarrow \varphi(\mathsf{z}))$, which trivally gives $T \vdash \delta \leftrightarrow \varphi(\overline{\ulcorner \delta \urcorner})$. □

Finally, a bit of jargon worth noting. By a certain abuse of mathematical terminology, we say

Defn. 52. *If δ is such that $T \vdash \delta \leftrightarrow \varphi(\overline{\ulcorner \delta \urcorner})$, then it is said to be a* fixed point *for φ.*

So the Diagonalization Lemma (or rather part (ii) of it) is often called the Fixed Point Theorem – every one-place open sentence has a fixed point.

10.2 Provability predicates

Recall our generalizing remark on p. 98. Suppose $Prf_T(m, n)$ is the relation which holds just if m is the super g.n. of a sequence of wffs that is a T proof of a sentence with g.n. n. This relation is p.r. decidable assuming T is p.r. axiomatized. Assuming T extends Q, T can capture any p.r. decidable relation, including Prf_T (§7.4). So we can legitimately stipulate

Defn. 53. $\mathsf{Prf}_T(\mathsf{x}, \mathsf{y})$ *stands in for a T-wff that canonically captures Prf_T,*

for there will indeed be such a wff. Next,

Defn. 54. *Put* $\mathsf{Prov}_T(\mathsf{y}) =_{\text{def}} \exists \mathsf{v}\, \mathsf{Prf}_T(\mathsf{v}, \mathsf{y})$*: such an expression is a (canonical) provability predicate for T.*

$\mathsf{Prov}_T(\overline{\mathsf{n}})$ is true, of course, on the standard arithmetic interpretation of T just if n numbers a T-theorem, i.e. a wff for which some number numbers a proof of it. Which means that $\mathsf{Prov}_T(\overline{\ulcorner\varphi\urcorner})$ is true just when φ is a theorem. Hence the aptness of the label **provability predicate** for Prov_T. Note, since Prov_T is built from a canonical Prf_T it is also Σ_1.

10.3 Incompleteness from the Diagonalization Lemma

Here is a general result about fixed points of such a provability predicate (as a reality check, ask yourself where subscript T's really belong in this statement and its proof):

Theorem 48. *Suppose T is p.r. axiomatized, contains Q, and some sentence or other γ is a fixed point for $\neg\mathsf{Prov}$ – i.e., $T \vdash \gamma \leftrightarrow \neg\mathsf{Prov}(\overline{\ulcorner\gamma\urcorner})$. Then (i) if T is consistent, $T \nvdash \gamma$. And (ii) if T is ω-consistent, $T \nvdash \neg\gamma$.*

Proof. (i) Suppose $T \vdash \gamma$. Then $T \vdash \neg\mathsf{Prov}(\overline{\ulcorner\gamma\urcorner})$. But if there *is* a proof of γ, then for some m, $Prf(m, \ulcorner\gamma\urcorner)$, so $T \vdash \mathsf{Prf}(\overline{\mathsf{m}}, \overline{\ulcorner\gamma\urcorner})$, since T captures Prf by Prf. Hence $T \vdash \exists\mathsf{x}\,\mathsf{Prf}(\mathsf{x}, \overline{\ulcorner\gamma\urcorner})$, i.e. we also have $T \vdash \mathsf{Prov}(\overline{\ulcorner\gamma\urcorner})$, making T inconsistent. So if T is consistent, $T \nvdash \gamma$.

(ii) Suppose $T \vdash \neg\gamma$. Then $T \vdash \mathsf{Prov}(\overline{\ulcorner\gamma\urcorner})$, i.e. $T \vdash \exists\mathsf{x}\,\mathsf{Prf}(\mathsf{x}, \overline{\ulcorner\gamma\urcorner})$. But given T is consistent, there is no proof of γ, i.e. for every m, not-$Prf(m, \ulcorner\gamma\urcorner)$, whence for every m, $T \vdash \neg\mathsf{Prf}(\overline{\mathsf{m}}, \overline{\ulcorner\gamma\urcorner})$. So we have a φ such that T proves $\exists\mathsf{x}\varphi(\mathsf{x})$ while it refutes each instance $\varphi(\overline{\mathsf{m}})$, which makes T ω-inconsistent. So if T is ω-consistent, $T \nvdash \neg\gamma$. □

But now note that the general Diagonalization Lemma implies as a special case

Theorem 49. *There exists a sentence γ such that $T \vdash \gamma \leftrightarrow \neg\mathsf{Prov}(\overline{\ulcorner\gamma\urcorner})$.*

Moreover, since Prov is Σ_1, $\neg\mathsf{Prov}$ is Π_1, and the diagonalization construction produces a Π_1 fixed point γ (see if you can work out why).

So putting those last two theorems together, we immediately recover Theorem 44.

Proving our old G_T is in fact a fixed point for $\neg\mathsf{Prov}_T$ Now our new proof and the old one are in fact very closely related, as explained in the book. Here we just note the following connection:

G_T, recall, is so constructed that it is true if and only if unprovable-in T. This fact can now be *expressed* inside T itself, by the wff $\mathsf{G}_T \leftrightarrow \neg\mathsf{Prov}_T(\overline{\ulcorner \mathsf{G_T} \urcorner})$. But T doesn't just express this fact but can quite easily *prove* it too, i.e. we have

Theorem 50. *If T is p.r. axiomatized and contains Q, $T \vdash \mathsf{G}_T \leftrightarrow \neg\mathsf{Prov}_T(\overline{\ulcorner \mathsf{G_T} \urcorner})$.*

In other words, G_T is one of those fixed points for $\neg\mathsf{Prov}_T$ which all have to be undecidable by T.

Proof. By definition, recall, $Gdl(m, n)$ holds iff $Prf(m, diag(n))$. We can therefore now retrospectively fix on the following canonical 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:

$\mathsf{G} =_{\text{def}} \forall \mathsf{x} \neg \mathsf{Gdl}(\mathsf{x}, \overline{\ulcorner \mathsf{U} \urcorner})$
$\leftrightarrow \quad \forall \mathsf{x} \neg \exists \mathsf{z}(\mathsf{Prf}(\mathsf{x}, \mathsf{z}) \wedge \mathsf{Diag}(\overline{\ulcorner \mathsf{U} \urcorner}, \mathsf{z}))$ (definition of Gdl)
$\leftrightarrow \quad \forall \mathsf{x} \forall \mathsf{z} \neg (\mathsf{Prf}(\mathsf{x}, \mathsf{z}) \wedge \mathsf{Diag}(\overline{\ulcorner \mathsf{U} \urcorner}, \mathsf{z}))$ (pushing in the negation)
$\leftrightarrow \quad \forall \mathsf{z} \forall \mathsf{x} \neg (\mathsf{Prf}(\mathsf{x}, \mathsf{z}) \wedge \mathsf{Diag}(\overline{\ulcorner \mathsf{U} \urcorner}, \mathsf{z}))$ (swapping quantifiers)
$\leftrightarrow \quad \forall \mathsf{z}(\mathsf{Diag}(\overline{\ulcorner \mathsf{U} \urcorner}, \mathsf{z}) \rightarrow \neg \exists \mathsf{x}\, \mathsf{Prf}(\mathsf{x}, \mathsf{z}))$ (rearranging after '$\forall \mathsf{z}$')
$\leftrightarrow \quad \forall \mathsf{z}(\mathsf{Diag}(\overline{\ulcorner \mathsf{U} \urcorner}, \mathsf{z}) \rightarrow \neg \exists \mathsf{v}\, \mathsf{Prf}(\mathsf{v}, \mathsf{z}))$ (changing variables)
$=_{\text{def}} \forall \mathsf{z}(\mathsf{Diag}(\overline{\ulcorner \mathsf{U} \urcorner}, \mathsf{z}) \rightarrow \neg \mathsf{Prov}(\mathsf{z}))$ (definition of Prov)

Since this is proved by simple logical manipulations, that means we can prove the equivalence inside the formal first-order logic built into Q and hence in T. So

$$T \vdash \mathsf{G} \leftrightarrow \forall \mathsf{z}(\mathsf{Diag}(\overline{\ulcorner \mathsf{U} \urcorner}, \mathsf{z}) \rightarrow \neg \mathsf{Prov}(\mathsf{z})).$$

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*t, it follows by definition that

$$T \vdash \forall \mathsf{z}(\mathsf{Diag}(\overline{\ulcorner \mathsf{U} \urcorner}, \mathsf{z}) \leftrightarrow \mathsf{z} = \overline{\ulcorner \mathsf{G} \urcorner}).$$

Putting those two results together, we immediately get

$$T \vdash \mathsf{G} \leftrightarrow \forall \mathsf{z}(\mathsf{z} = \overline{\ulcorner \mathsf{G} \urcorner} \rightarrow \neg \mathsf{Prov}(\mathsf{z})).$$

But the right-hand side of that biconditional is trivially equivalent to $\neg \mathsf{Prov}(\overline{\ulcorner \mathsf{G} \urcorner})$. So we've proved the desired result. □

10.4 Tarski's Theorem

In a way, Rosser's Theorem which – as it were – tidies up the First Theorem by enabling us to get rid of the assumption of ω-consistency is the natural next topic to look at. And that's what I do in the book. But here let's proceed in a different order and next 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*.

10.4.1 Truth-predicates and truth-definitions

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 sensible instance of *'φ' is true iff* φ is true. And that's because of the meaning of the informal truth-predicate 'true'.

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.

Now suppose, just suppose, we have some expression $\mathsf{T}(\mathsf{x})$ which is suitably defined to expresses this numerical property *True*, and let L' be the result of adding $\mathsf{T}(\mathsf{x})$ to our initial language L. (For the moment, we leave it open whether L' is just L, which it would be if $\mathsf{T}(\mathsf{x})$ is in fact already definable from L's resources.)

Then, by definition, for any L-sentence φ, we have

φ is true iff *True*$(\ulcorner \varphi \urcorner)$ iff $\mathsf{T}(\overline{\ulcorner \varphi \urcorner})$ is true.

Hence, for any L-sentence φ, every corresponding L'-sentence

$$\mathsf{T}(\overline{\ulcorner\varphi\urcorner}) \leftrightarrow \varphi$$

is true. Which motivates our first main definition:

Defn. 55. *An open L'-wff $\mathsf{T}(\mathsf{x})$ is a formal* **truth-predicate** *for L iff for every L-sentence φ, $\mathsf{T}(\overline{\ulcorner\varphi\urcorner}) \leftrightarrow \varphi$ is true.*

And here's a companion definition:

Defn. 56. *A theory Θ (with language L' which includes L) is a* **truth-theory** *for L iff for some L'-wff $\mathsf{T}(\mathsf{x})$, $\Theta \vdash \mathsf{T}(\overline{\ulcorner\varphi\urcorner}) \leftrightarrow \varphi$ for every L-sentence φ.*

Equally often, a truth-theory for L is called a 'definition of truth for L'.

So: in sum, a truth-predicate T is a predicate that applies to (the Gödel numbers for) true sentences, and so *expresses* truth, and a truth-theory is a theory which *proves* the right T-biconditionals.

10.4.2 The undefinability of truth

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 51. *No consistent theory T which includes Q can define truth for its own language.*

Proof. Assume T defines truth for L, i.e. there is an L-predicate $\mathsf{T}(\mathsf{x})$ such that $T \vdash \mathsf{T}(\overline{\ulcorner\varphi\urcorner}) \leftrightarrow \varphi$ for every L-sentence φ. Since T is has the right properties, the Diagonalization Lemma applies, so applying the Lemma to $\neg\mathsf{T}(\mathsf{x})$, we know that there must be some sentence L – a Liar sentence! – such that

1. $T \vdash \mathsf{L} \leftrightarrow \neg\mathsf{T}(\overline{\ulcorner\mathsf{L}\urcorner})$.

But, by our initial assumption, we also have

2. $T \vdash \mathsf{T}(\overline{\ulcorner\mathsf{L}\urcorner}) \leftrightarrow \mathsf{L}$.

It is immediate that T is inconsistent, contrary to hypothesis. So our assumption must be wrong: T can't define truth for its own language. □

10.4.3 The inexpressibility of truth

That first theorem puts limits on what a nice theory can *prove* about truth. But we can also put limits on what a theory's language can even *express* about truth.

Consider our old friend L_A for the moment, and suppose that there is an L_A truth-predicate T_A that expresses the corresponding truth property $True_A$. The first part of the Diagonalization Lemma for the theory Q applies, in particular to the negation of $\mathsf{T}_A(\mathsf{x})$. So we know that for some L_A sentence L,

1. $\mathsf{L} \leftrightarrow \neg\mathsf{T}_A(\overline{\ulcorner \mathsf{L} \urcorner})$.

is true. But, by the assumption that T_A is a truth-predicate for L_A,

3. $\mathsf{T}_A(\overline{\ulcorner \mathsf{L} \urcorner}) \leftrightarrow \mathsf{L}$

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 the first part of the Diagonalization Lemma again to be provable. Call that an arithmetically adequate language. Then by the same argument,

Theorem 52. *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 truth) inside the theory.

The Master Argument for incompleteness Our results about the non-expressibility of truth of course point to a 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 (by a provability predicate), truth-in-L_A *isn't* expressible. So assuming that PA is sound so that 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 true roots of the phenomenon. 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.

In sum, as we emphasized before, arithmetical truth and provability in this or that formal system must peel apart.

How does that statement of Gödel's square the importance that he placed on the syntactic version of the First Theorem? Well, Gödel was a realist about mathematics: *he* believed in the real existence of mathematical entities, and believed that our theories (at least aim to) deliver truths about them. But that wasn't the dominant belief among those around him concerned with foundational matters. As I put it before, for various reasons (for a start, think logical positivism), the very idea of truth-in-mathematics was under some suspicion at the time. So even though semantic notions were at the root of Gödel's insight, it was extremely important – given the intended audience – to show that you don't need to deploy those semantic notions to prove incompleteness.

10.5 Rosser's Theorem

10.5.1 Rosser's basic trick

One half of the First Theorem requires the assumption that we are dealing with a theory T which is not only consistent 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 so-called '1-consistency' in showing that $T \nvdash \neg\mathsf{G}_T$.

2. Following Barkley Rosser, 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, that's the result we'll talk about here (I say something about 1-consistency in the book).

So 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 'Rosser 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).

10.5.2 Implementing the trick

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 has the usual properties it will be captured by a wff $\overline{\mathsf{Prf}}_T(\mathsf{x}, \mathsf{y})$. So let's consider *the Rosser provability predicate* defined as follows:

Defn. 57. $\mathsf{RProv}_T(\mathsf{x}) =_{\text{def}} \exists\mathsf{v}(\mathsf{Prf}_T(\mathsf{v}, \mathsf{x}) \wedge (\forall\mathsf{w} \leq \mathsf{v})\neg\overline{\mathsf{Prf}}_T(\mathsf{w}, \mathsf{x}))$.

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.

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 the Rosser provability predicate. The Lemma then tells us,

Theorem 53. *Given that T is p.r. axiomatized and contains Q, then there is a sentence R_T such that $T \vdash \mathsf{R}_T \leftrightarrow \neg\mathsf{RProv}_T(\ulcorner\mathsf{R}_T\urcorner)$.*

We call such a sentence R_T a Rosser sentence for T.

10.5.3 Rosser's Theorem

We can now show that

Theorem 54. *Let T be consistent p.r. axiomatized theory including Q and let ρ be any fixed point for $\neg\mathsf{RProv}_T(\mathsf{x})$. Then $T \nvdash \rho$ and $T \nvdash \neg\rho$.*

And since the Diagonalization Lemma tells us that there is a fixed point, it follows that T has an undecidable sentence R_T, without now requiring ω-consistency. Sadly, however – and there's no getting away from it – the proof of Theorem 54

is messy and very unpretty. Masochists can check out the proof of Theorem 25.4 in *IGT2*. We then have to do more work to beef up that proof idea to show that in fact (as with Gödel's original proof) we can find a Π_1 sentence which is undecidable so long as T is consistent. You do not need to know these proofs! – just that they exist, so we get Rosser's Theorem (compare Theorem 45):

Theorem 55. *If T is a p.r. adequate, p.r. axiomatized theory whose language includes L_A, then there is Π_1 sentence φ such that, if T is consistent then $T \nvdash \varphi$ and $T \nvdash \neg\varphi$.*

Further reading

And that's enough – at least in these notes – about the First Incompleteness Theorem. There's quite a bit more in the book, in Chs 21–30, which I'm not going to be covering here. Enthusiasts will doubtless want to devour the lot, but let me especially highlight the sections which amplify this episode, and then other the sections you ought to know about for general logical purposes anyway:

1. More on the topics in the episode: Chs 24, 25, 27.
2. For some generalizing of the scope of the incompleteness theorem see Ch. 26. For something on second-order arithmetics, see §§29.1–22.6.

Episode 11

The Second Incompleteness Theorem

- Con_T, a canonical consistency sentence for T
- The Formalized First Theorem
- The Second Theorem
- Why the Second Theorem matters
- What it takes to prove it
- Con_T as a Gödel sentence
- Theories that 'prove' their own inconsistency

We have said all that we are going to say in these notes about the *First* Incompleteness Theorem. The previous episode started with a detailed review of how we got to that theorem, and we don't need so pause for another detailed review of where we have been. We will therefore just dive straight into our discussion of the *Second* Incompleteness Theorem.

11.1 Defining Con_T

We haven't put any requirement on the particular formulation of first-order logic built into Q (and hence any theory which contains it). In particular, the logic may or may not have a built-in **absurdity constant** like $\bot$. But henceforth, let's use the sign '$\bot$' in the following way:

Defn. 58. *'$\bot$' is T's built-in absurdity constant if it has one, or else it is an abbreviation for '$\mathsf{0} = \overline{\mathsf{1}}$'.*

If T contains Q, T of course proves $\mathsf{0} \neq \overline{\mathsf{1}}$. So on either reading of '$\bot$', if T also proves $\bot$, it is inconsistent. And if T's has a standard classical (or indeed intuitionistic) logic and T is inconsistent, then on either reading it will prove $\bot$.

Now recall these two definitions:

Defn. 53. $\mathsf{Prf}_T(\mathsf{x}, \mathsf{y})$ *stands in for a T-wff that canonically captures Prf_T,*

Defn. 54. *Put* $\mathsf{Prov}_T(\mathsf{y}) =_{\text{def}} \exists \mathsf{v}\, \mathsf{Prf}_T(\mathsf{v}, \mathsf{y})$*: such an expression is a (canonical) provability predicate for T.*

The wff $\neg\mathsf{Prov}_T(\overline{\ulcorner \bot \urcorner})$ will then be true if and only if T *doesn't* prove $\bot$, i.e. (given what we've just said) if and only if T is consistent. That evidently motivates the definition

Defn. 59. Con_T *abbreviates* $\neg\mathsf{Prov}_T(\overline{\ulcorner \bot \urcorner})$.

For obvious reasons, the arithmetic sentence Con_T is called a (canonical) **consistency sentence** for T.

Or at least, Con_T is the crispest definition of a consistency sentence for T. There are natural alternatives. Suppose $\mathsf{Contr}(\mathsf{x}, \mathsf{y})$ captures the p.r. relation which holds between two numbers when they code for a contradictory pair of sentences, i.e. one codes for some sentence φ and the other for $\neg\varphi$. Then we could put

Defn. 60. $\mathsf{Con}'_T =_{\text{def}} \neg\exists \mathsf{x} \exists \mathsf{y}(\mathsf{Prov}_T(\mathsf{x}) \wedge \mathsf{Prov}_T(\mathsf{y}) \wedge \mathsf{Contr}(\mathsf{x}, \mathsf{y}))$,

and this would be another natural way of expressing T's consistency. But, on modest assumptions, this second definition and its variants gives us a wff which is provably equivalent to Con_T: so we'll stick to our standard definition.

Finally, note that since Prov_T is Σ_1, Con_T is Π_1.

11.2 From the Formalized First Theorem to the Second Theorem

Assume in this section that we are dealing with a theory T which is p.r. axiomatized and contains Q. Then one half of the First Incompleteness Theorem tells us that

(1) If T is consistent then G_T is not provable in T.

(Remember, we only need the idea of ω-consistency for the *other* half of the First Theorem).

We now have a natural way of expressing (1) inside the formal theory T itself, i.e. by

(2) $\mathsf{Con}_T \to \neg\mathsf{Prov}_T(\overline{\ulcorner \mathsf{G}_T \urcorner})$.

Pause next to reflect that the informal reasoning for the First Theorem is in fact rather elementary (we certainly needed no higher mathematics at all, just simple reasoning about arithmetical matters). So we might well expect that if T contains enough arithmetic – maybe a bit more than is given by Q, e.g. maybe a bit of induction – it should itself be able to replicate that elementary reasoning.

In other words, if T is arithmetically strong enough, then T will not only be able to express (half) of the First Theorem via the wff abbreviated (2), but should be able to prove it too! Hence, for suitably strong T we'd hope to have

Theorem 56. $T \vdash \mathsf{Con}_T \to \neg\mathsf{Prov}_T(\overline{\ulcorner \mathsf{G}_T \urcorner})$.

We can call such a result the **Formalized First Theorem** for the relevant provability predicate. (Of course, we'll want to fill this out – what counts as a suitable strong theory T? – but we'll do that in a moment.)

However, we already have

Theorem 50. *If T is p.r. axiomatized and contains Q, $T \vdash \mathsf{G}_T \leftrightarrow \neg\mathsf{Prov}_T(\overline{\ulcorner \mathsf{G}_T \urcorner})$.*

So we can then immediately infer from these two theorems that, for suitably strong T,

(3) if $T \vdash \mathsf{Con}_T$, then $T \vdash \mathsf{G}_T$.

But we know from the First Theorem that,

(4) If T is consistent, $T \nvdash \mathsf{G}_T$.

So the Formalized First Theorem immediately yields the unprovability of the relevant consistency sentence. Hence, putting what we have so far, we get

Theorem 57. *Supposed T is p.r. axiomatized and contains enough arithmetic: then if T is consistent, then $T \nvdash \mathsf{Con}_T$.*

Which is a somewhat unspecific version of the **Second Incompleteness Theorem**.

Historical note In his original 1931 paper, Gödel just states a version of the Second Theorem, again relying on the arm-waving thought that the reasoning for the First Theorem is so elementary that a strong enough theory (like the theory P he officially considers) must be able to replicate the reasoning and so prove the Formalized version of the theorem. He too didn't spell out the details.

The hard work of taking a strong enough T and showing it proves the Formalized First Theorem was first done for a particular case by David Hilbert and Paul Bernays in their *Grundlagen der Mathematik* of 1939. The details of their proof are – the story goes – due to Bernays, who had discussed it with Gödel during a transatlantic voyage.

Now, Hilbert and Bernays helpfully isolated '**derivability conditions**' on the predicate Prov_T, conditions whose satisfaction is indeed enough for a theory T to prove the Formalized First Theorem. Later, Martin H. Löb gave a rather neater version of these conditions: and it is his version which has become standard and features in our discussion below. These derivability conditions are consequently often called the HBL conditions.

11.3 How interesting is the Second Theorem?

You might well think: 'OK, so we can't derive Con_T in T. But that fact is of course no evidence at all *against* T's consistency, since we already know from the First Theorem that various true claims about unprovability – like the standardly constructed Gödel sentence G_T – will be underivable in T. On the other hand, if – *per impossibile* – we could have given a T proof of Con_T, that wouldn't have given us any special evidence *for* T's consistency: we could simply reflect that even if T were inconsistent we'd still be able to derive Con_T, since we can derive *anything* in an inconsistent theory! Hence the derivability or otherwise of

a canonical statement of T's consistency inside T itself can't show us a great deal.'

But, on reflection, the Theorem *does* yield some plainly important and substantial corollaries, of which the most important is this:

Theorem 58. *Suppose S is a consistent theory, strong enough for the Second Theorem to apply to it, and W is a weaker fragment of S, then $W \nvdash \mathsf{Con}_S$.*

That's because, if S can't prove Con_S, because of the Second Thereom, then a fortiori *part* of S can't prove it .

So, for example, we *can't* take some problematic strong theory like set theory which (radically) extends arithmetic and show that it is consistent by (i) using arithmetic coding for talking about its proofs and then (ii) using uncontentious reasoning already available in some relatively weak, purely arithmetical, theory.

Which means that the Second Theory – at least at first blush – sabotages Hilbert's Programme (see p. 93, and the longer discussions in *IGT*2).

11.4 Sharpening the Second Theorem

$\mathsf{Prov}_T(\mathsf{y})$ abbreviates $\exists \mathsf{v}\, \mathsf{Prf}_T(\mathsf{v}, \mathsf{y})$; and Prf_T is a Σ_1 expression. So arguing about provability inside T will involve establishing some general claims involving Σ_1 expressions. And how do we prove quantified claims? Using induction is the default method.

It therefore looks quite a good bet that T will itself be able to prove (the relevant half of) the First Theorem for T, i.e. we will have $T \vdash \mathsf{Con}_T \rightarrow \neg\mathsf{Prov}_T(\ulcorner \overline{\mathsf{G}_T} \urcorner)$, if T at least 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 no more complex than Σ_1. So let's define

Defn. 61. *A theory is Σ-normal if and only if it is p.r. axiomatized, contains Q, and also includes induction at least for Σ_1 wffs.*

PA is of course Σ-normal. (Warning: 'Σ-normal' is my shorthand: there seems to be no standard term here.)

Then the following looks a plausible conjecture

Theorem 59. *If T is Σ-normal, then T proves the formalized First Theorem,*

and then if T is consistent, $T \nvdash \mathsf{Con}_T$. That sharpens our vaguely stated Theorem 57; and this sharp version of the Formalized First Theorem is indeed provable. We won't give a full proof, however: but in the next-but-one section, we'll

say something about how the details get filled in. But first we introduce some standard notation.

11.5 The box notation

To improve readability, let's introduce some standard notation. We will henceforth abbreviate $\mathsf{Prov}_T(\ulcorner\varphi\urcorner)$ simply by $\Box_T\varphi$. This is perhaps a bit naughty, as the suggested new notation hides some Gödel-numbering, which is like hiding quotation marks. But it is safe enough if we keep our wits about us.

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's *The Logic of Provability.*

So in particular, $\neg\mathsf{Prov}_T(\ulcorner\mathsf{G}_T\urcorner)$ can be abbreviated $\neg\Box_T\mathsf{G}_T$. Thus in our new notation, the Formalized First Theorem is $T \vdash \mathsf{Con}_T \rightarrow \neg\Box_T\mathsf{G}_T$. Moreover, Con_T can now alternatively be abbreviated as $\neg\Box_T\bot$.

However, we will very often drop the explicit subscript T from the box symbol and elsewhere, and let context supply it.

11.6 Proving the Formalized First Theorem

First a standard definition: we will say (dropping subscripts)

Defn. 62. *The derivability conditions hold in T if and only if, for any T-sentences φ, ψ,*

C1. If $T \vdash \varphi$, then $T \vdash \Box\varphi$,

C2. $T \vdash \Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$,

C3. $T \vdash \Box\varphi \rightarrow \Box\Box\varphi$.

We can then prove

Theorem 60. *If T is Σ-normal, then the derivability conditions hold for T.*

Now, proving this in detail is a seriously tedious task, and I certainly don't propose to do it here, nor do I do it completely in the current edition of *IGT* either. But suppose we've done the hack work needed to nail down all the details, and have proved that theorem. Then we can go on to show that

Theorem 61. *If T is Σ-normal and the derivability conditions hold for T, then T proves the Formalized First Theorem.*

These two theorems together evidently entail Theorem 59.

We'll now prove Theorem 61, as this is just a mildly fun exercise in box-pushing. The target is prove, on the given assumptions, that $T \vdash \mathsf{Con} \rightarrow \neg\Box\mathsf{G}$.

Proof. First, since T is p.r. axiomatized and contains Q, Theorem 50 holds. So, in our new symbolism, we have $T \vdash \mathsf{G} \leftrightarrow \neg\Box\mathsf{G}$.

Second, note that for any theory T containing Q, $T \vdash \neg\bot$ (either by the built-in logic, or because we've put $\bot =_{\text{def}} \mathsf{0} = \overline{1}$). And simple logic will show that, for any wff φ, we have

$$T \vdash \neg\varphi \rightarrow (\varphi \rightarrow \bot).$$

Given the latter and the derivability condition (C1), this means

$$T \vdash \Box(\neg\varphi \rightarrow (\varphi \rightarrow \bot)).$$

So given the derivability condition (C2) and using modus ponens, it follows that for any φ

A. $T \vdash \Box\neg\varphi \rightarrow \Box(\varphi \rightarrow \bot)$.

We now argue as follows:

1.	$T \vdash \mathsf{G} \rightarrow \neg\Box\mathsf{G}$	Half of Thm 50
2.	$T \vdash \Box(\mathsf{G} \rightarrow \neg\Box\mathsf{G})$	From 1, given C1
3.	$T \vdash \Box\mathsf{G} \rightarrow \Box\neg\Box\mathsf{G}$	From 2, given C2
4.	$T \vdash \Box\neg\Box\mathsf{G} \rightarrow \Box(\Box\mathsf{G} \rightarrow \bot)$	Instance of A
5.	$T \vdash \Box\mathsf{G} \rightarrow \Box(\Box\mathsf{G} \rightarrow \bot)$	From 3 and 4
6.	$T \vdash \Box\mathsf{G} \rightarrow (\Box\Box\mathsf{G} \rightarrow \Box\bot)$	From 5, given C2
7.	$T \vdash \Box\mathsf{G} \rightarrow \Box\Box\mathsf{G}$	Instance of C3
8.	$T \vdash \Box\mathsf{G} \rightarrow \Box\bot$	From 6 and 7
9.	$T \vdash \neg\Box\bot \rightarrow \neg\Box\mathsf{G}$	Contraposing
10.	$T \vdash \mathsf{Con} \rightarrow \neg\Box\mathsf{G}$	Definition of Con

Which gives us the Formalized First Theorem (and hence, as before, the Second Incompleteness Theorem). □

11.7 Other excitements

We'll assume throughout this section that T is such that the derivability conditions hold (Σ-normality suffices, see Def. 61).

11.7.1 G_T and Con_T are provably equivalent in T

We know from Theorem 61 that $T \vdash \mathsf{Con} \to \neg\Box\mathsf{G}$ (suppressing subscripts!).

Now note the following result which says that T knows that, *if* it can't prove φ, it must be consistent.

Theorem 62. *For any sentence φ, $T \vdash \neg\Box\varphi \to \mathsf{Con}$.*

Proof. We argue as follows:

1.	$T \vdash \bot \to \varphi$	Logic!
2.	$T \vdash \Box(\bot \to \varphi)$	From 1, given C1
3.	$T \vdash \Box\bot \to \Box\varphi$	From 2, given C2
4.	$T \vdash \neg\Box\varphi \to \neg\Box\bot$	Contraposing
5.	$T \vdash \neg\Box\varphi \to \mathsf{Con}$	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 the usual sort of theory: 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.

Now, as a particular instance of this, we have $T \vdash \neg\Box\mathsf{G} \to \mathsf{Con}$. So putting that together with Theorem 61, we have $T \vdash \mathsf{Con} \leftrightarrow \neg\Box\mathsf{G}$. And now combine *that* with Theorem 50 which tells us that $T \vdash \mathsf{G} \leftrightarrow \neg\Box\mathsf{G}$, and low and behold we've shown

Theorem 63. *If T is Σ-normal then $T \vdash \mathsf{Con} \leftrightarrow \mathsf{G}$.*

This means that, not only do we have $T \nvdash \mathsf{Con}$, we also have (assuming T is ω-consistent) $T \nvdash \neg\mathsf{Con}$. In other words, Con is formally undecidable by T.

But Con is *not* self-referential in any way, however loosely interpreted. That observation should scotch once and for all any lingering suspicion that the incompleteness phenomena are somehow inevitably tainted by self-referential paradox.

11.7.2 Con is a fixed point of the provability predicate

We now prove

Theorem 64. $T \vdash \mathsf{Con} \leftrightarrow \neg\Box\mathsf{Con}$.

Proof The direction from right to left is an instance of Theorem 62. For the other direction, note that

1.	$T \vdash \mathsf{Con} \rightarrow \mathsf{G}$	Already proved
2.	$T \vdash \Box(\mathsf{Con} \rightarrow \mathsf{G})$	From 1, given C1
3.	$T \vdash \Box\mathsf{Con} \rightarrow \Box\mathsf{G}$	From 2, given C2
4.	$T \vdash \neg\Box\mathsf{G} \rightarrow \neg\Box\mathsf{Con}$	Contraposing
5.	$T \vdash \mathsf{Con} \rightarrow \neg\Box\mathsf{Con}$	Using Thm. 61

So: this shows that Con (like G) is also a fixed point of the provability predicate.

Some authors refer to any fixed point of the provability predicate as a Gödel sentence. Fine. That's one way of using the jargon. But if you adopt the broad usage, you must be careful with your informal commentary. For example, not all Gödel sentences in the broad sense 'say' *I am unprovable*: Con is a case in point.

Theories that 'prove' their own inconsistency An ω-consistent T can't prove $\neg\mathsf{Con}_T$, as we've just noted. 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,

Theorem 65. *Under the given conditions,* $\vdash \neg\mathsf{Con}_S \rightarrow \neg\mathsf{Con}_R$.

Proof. Suppose $\neg\mathsf{Con}_S$, i.e. suppose $\exists \mathsf{v}\, \mathsf{Prf}_S(\mathsf{v}, \perp)$. Hence for some a, $\mathsf{Prf}_S(\mathsf{a}, \perp)$. And that implies $\mathsf{Prf}_R(\mathsf{a}, \perp)$. 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 $\exists \mathsf{v}\, \mathsf{Prf}_R(\mathsf{v}, \perp)$, i.e. $\neg\mathsf{Con}_R$. And the inferences here only use first-order logic. □

Now let's put that theorem to use. Take the simpler theory S to be PA (which is Σ-normal!). 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? because that means $R \vdash \neg\mathsf{G}_R$, which it wouldn't be the case if R were ω-consistent). 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,

Theorem 66. *Assuming* PA *is consistent, the theory* $R = \mathsf{PA} + \neg\mathsf{Con}_{\mathsf{PA}}$ *is a consistent theory which 'proves' its own inconsistency.*

And since R proves $\neg\mathsf{Con}_R$,

Theorem 67. *The* consistent *theory* R *is 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.

Now, the derivability conditions hold for theories that contain PA, so they will hold for R. Hence by Theorem 63, $R \vdash \mathsf{Con}_R \leftrightarrow \mathsf{G}_R$. 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 the cut-down requirement of 1-consistency explained in the book – *has* to be assumed in the proof that arithmetic is incomplete, if we are to prove it by constructing an original-style Gödel sentence like G_R.

Further reading

There's a lot more in the book around and about these issues, in Chs 31–37. If you are going to read more, however, perhaps the three things to concentrate on are these, which push things on a bit. First, the discussion of Hilbert's Programme

in the first half of Chap. 37 (as the topic is historically important). Second, the quick remarks about minds and machines later in Chap. 37. And then – for enthusiasts – there are the more intricate arguments in Chap. 36 leading up to discussion of what Gödel called the 'best and more general version' of the Second Theorem.

Episode 12

Curry's Paradox and Löb's Theorem

- Curry's Paradox
- Curry's Theorem
- Löb's Theorem
- Löb's Theorem implies the Second Incompleteness Theorem again
- What's still to come ...

This final episode is really by way of dessert (and to add to the fun, the discussion proceeds a little differently to the treatment of these topics in *IGT2*). We start by introducing Curry's Paradox, and we then use the line of reasoning that leads to this paradox to prove a *theorem*, which we can call Curry's Theorem. We'll draw out a couple of significant corollaries of this theorem

Then we locate some variant reasoning that generates a version of Curry's Paradox which uses weaker assumptions about truth, and we show that *this* reasoning can be mirrored by reasoning about the provability predicate for any theory which contains enough arithmetic, and that yields Löb's Theorem.

We'll see that Löb's Theorem immediately entails the Second Incompleteness theorem (and indeed, vice versa).

We end with a quick overview of where the rest of *IGT2* goes, but we won't follow the story any further in these notes.

12.1 Curry's Paradox: How to show that the moon is made of green cheese

We seemingly can construct a sentence that says e.g. 'if I am true, then the moon is made of green cheese' (after all, haven't I just constructed one?). Let's symbolize such a sentence by simply δ, and let's put φ for 'the moon is made of green cheese'.

Then, by definition, δ holds just in case, if δ is true, then φ. We therefore have the following: $\delta \leftrightarrow (\text{True}(\delta) \rightarrow \varphi)$. Assuming the usual equivalence between δ and δ's being true – an assumption we'll revisit in due course – that's equivalent to $\delta \leftrightarrow (\delta \rightarrow \varphi)$. And now we can argue as follows (and because I'm among logical friends, I'll regiment the argument Fitch-style: but do remember that – for the moment – this is just supposed to perspicuously set out *informal* reasoning about an informal claim).

1	$\delta \leftrightarrow (\delta \rightarrow \varphi)$	Given, by definition(!?)
2	δ	Supposition
3	$\delta \rightarrow (\delta \rightarrow \varphi)$	From biconditional 1
4	$\delta \rightarrow \varphi$	Modus Ponens from 2, 3
5	φ	Modus Ponens from 2, 4
6	$\delta \rightarrow \varphi$	Conditional Proof from subproof 2 to 5
7	$(\delta \rightarrow \varphi) \rightarrow \delta$	From biconditional 1
8	δ	Modus Ponens from 6, 7
9	φ	Modus Ponens from 6, 8

So the moon is indeed made of green cheese!

This sort of argument was known to medieval logicians, but was rediscovered by Haskell Curry in 1942, hence **Curry's Paradox**.

Why 'paradox'? Because on the face of it truths of the kind (1) seem to be available to us, given we can construct self-referential sentences (surely often

harmless, as in 'This sentence contains five words'). Yet evidently something has gone wrong.

Can we pin the blame on anything *after* the starting assumption at line (1)? At least at first sight, seemingly not, for we have only used intuitively secure reasoning about the conditional: there is, for example, no 'vacuous discharge' or other dodgy move.[1] True, we do use the supposition at line (2) and also the conditional at line (6) twice; but why should *that* matter?[2]

So, at least at first sight, the blame seems to fall squarely on the original assumption that there is such a δ as makes (1) hold. That's perhaps no great surprise. For the claim 'if I am true, then the moon is made of green cheese', seems to be in the same ball-park as the liar statement 'I am *not* true'. And we all know the sort of trouble that *that* gets us into! (Though also we know how difficult it is to give principled reasons for avoiding such trouble.)

Still, there *is* something new being revealed by Curry's Paradox, namely that there is a paradox of intuitively the same general kind as the Liar Paradox, but which does not explicitly involve negation. So even if boxing clever with negation might provide a response to the Liar Paradox (we might go dialethic, for example, and allow a proposition and its negation sometimes to be true together), this sort of approach isn't going to immediately transmute into a way of dealing with Curry's Paradox. And we might reasonably think that *that* is an initial strike against trying to deal with the Liar just by getting fancy with negation.

12.2 Curry's theorem, and two implications

Let's now go more formal, and turn the informal reasoning for a paradox in the last section into formal reasoning for a theorem in this section.

So, we will say that a theory T *has a standard conditional* (and associated standard biconditional) if the reasoning in our displayed proof in the previous section goes through, once we grant the given assumption at the first line. In other words, T allows Modus Ponens, non-vacuous Conditional Proof, and doesn't restrict the re-use of data. Then we evidently have the following formal result we

[1] Vacuous discharge? Standard natural deduction systems allow us to take a subproof from the temporary assumption ψ to conclusion φ, discharge the assumption, and infer $\psi \to \varphi$ – even when ψ isn't actually used in getting to φ.

[2] OK, there are restricted systems of substructural logics that care about the re-use of data, for that sort of thing can matter in computer science. But it is one thing to be interested in such formal systems because we want, e.g., ways of keeping track of the re-use of data, and another thing entirely to suppose that such re-use is actually fallacious.

can call Curry's Theorem (I'd better remark that this is *not* a standard label, and indeed the Theorem isn't often spelt out like this).

Theorem 68. *If theory T has a standard conditional, and if for every sentence φ of T's language there is some corresponding sentence δ such that $T \vdash \delta \leftrightarrow (\delta \rightarrow \varphi)$, then T is inconsistent.*

For if the condition of this theorem is satisfied, then for all φ, T proves some $\delta \leftrightarrow (\delta \rightarrow \varphi)$ and hence – by a formalized version of the reasoning given above using T-s proof-system – T proves φ. And if it proves every sentence φ, T is inconsistent.

Let's draw two conclusions from this abstract result. Both are closely related to things you already know. Still, can often be illuminating to get at variants of old conclusions via new routes.

So first let's apply Curry's Theorem to set theory. The **naive comprehension** principle, recall, says that

> For any monadic condition $\chi(x)$, $\exists_1 y \forall x(x \in y \leftrightarrow \chi(x))$.

Now, we already know that naive comprehension leads to trouble via Russell's Paradox in a theory equipped with a sensible negation (just put $\chi(x) =_{\text{def}} \neg x \in x$) and contradiction rapidly follows). But Curry's Theorem shows that naive comprehension still leads to trouble however we doctor the negation rules, so long as the conditional remains standard:

Theorem 69. *If T is a set theory with a standard conditional and naive comprehension, it is inconsistent.*

Proof. Choose φ as you like, and then put $\chi(x) =_{\text{def}} (x \in x \rightarrow \varphi)$. Applying naive comprehension, there is a set we can dub τ such that $T \vdash \forall x(x \in \tau \leftrightarrow (x \in x \rightarrow \varphi))$. So, in particular, $T \vdash \tau \in \tau \leftrightarrow (\tau \in \tau \rightarrow \varphi)$. Put δ for $\tau \in \tau$ and we've shown that for any φ you choose, there's a δ such that T proves $\delta \leftrightarrow (\delta \rightarrow \varphi)$. So T is inconsistent by Curry's Theorem. □

Our next application takes us back to issues about truth again. Recall the Diagonalization Lemma, Theorem 47. Suppose, then, that T includes Q, so that for every one-place open sentence $\varphi(x)$ of T's language, there is some fixed point sentence δ such that $T \vdash \delta \leftrightarrow \varphi(\overline{\ulcorner \delta \urcorner})$ (where as before, $\overline{\ulcorner \delta \urcorner}$ is T's numeral for $\ulcorner \delta \urcorner$, the Gödel number of δ on some sensible coding scheme).

Let's say that T 'has a naive theory of truth' if we can form an open sentence $\mathsf{T}(x)$ in T such that $T \vdash \mathsf{T}(\overline{\ulcorner \sigma \urcorner}) \leftrightarrow \sigma$ unrestrictedly, for every sentence σ (i.e. all 'T-biconditionals' are provable).

Earlier we proved Tarski's theorem in the form

Theorem 51. *No consistent theory T which includes Q can define truth for its own language.*

Which says, precisely, that T cannot have a naive theory of truth. Here now is another version Tarski's theorem:

Theorem 70. *If T is consistent, has a standard conditional and includes Q, it can't also have a naive theory of truth.*

Proof. Assume for reductio that T has a naive theory of truth. Choose φ as you like, and consider the one-place open sentence $\mathsf{T}(x) \to \varphi$. The Diagonalization Lemma tells us that for some δ, T proves $\delta \leftrightarrow (\mathsf{T}(\overline{\ulcorner \delta \urcorner}) \to \varphi)$. By the assumption that $T \vdash \mathsf{T}(\overline{\ulcorner \delta \urcorner}) \leftrightarrow \delta$, it follows that T proves $\delta \leftrightarrow (\delta \to \varphi)$. So by Curry's Theorem, T would be inconsistent. □

In sum, you can't get a consistent naive set theory, or consistently add a naive treatment of truth to a theory which knows about arithmetic, just by boxing clever with negation and/or swallowing certain contradictions and banning the "arguing past contradictions" which produces explosion. Even if we fancy-up our treatment of negation, so long as our theory has a standard conditional (i.e. one that allows Modus Ponens, non-vacuous Conditional Proof, and doesn't restrict the re-use of data), that is damaging enough. As Geach remarked long ago when commenting on a Curry-like argument in his 1955 *Analysis* note 'On "insolubilia"' – a title that reminds us that there are medieval antecedents to these discussions –

> If we want to retain the naive view of truth, or the naive view of classes ..., then we must modify the elementary rules of inference relating to 'if'.

12.3 Proving the moon is made of green cheese again and Löb's Theorem

Let's now, for brevity's sake, rewrite '$\mathsf{T}(\overline{\ulcorner \delta \urcorner})$' as '$\top\delta$' (compare our use of the box in the last episode).

Now, at the outset, we informally used the usual naive equivalence between δ and δ's being true in proposing to regiment 'if I am true, then φ' as a sentence δ such that $\delta \leftrightarrow (\delta \to \varphi)$. And the derivation of our last theorem depends

on the formal reflection of that equivalence, i.e. the principle that (for any δ) $T \vdash \top\delta \leftrightarrow \delta$. But in fact, we don't the need that equivalence – we don't need the full content of the 'naive theory of truth' – to get a version of Curry's paradox.

To explain, suppose we are dealing with a predicate T in a theory T for which the following principles hold (schematically in φ, ψ):[3]

$\top 0$. $T \vdash \top\varphi \rightarrow \varphi$,

$\top 1$. $T \vdash \top(\varphi \rightarrow \psi) \rightarrow (\top\varphi \rightarrow \top\psi)$,

$\top 2$. $T \vdash \top\varphi \rightarrow \top\top\varphi$,

$\top 3$. If $T \vdash \varphi$ then $T \vdash \top\varphi$.

Now let's again put φ for 'the moon is made of green cheese', and let δ say of itself 'if I am true, then the moon is made of green cheese', so we have $\delta \leftrightarrow (\top\delta \rightarrow \varphi)$. We again argue to the conclusion φ, but this time using less than a full naive truth-theory but just assuming we are entitled to the principles we've just given.

[3] Aside for modal logic enthusiasts: these principles are parallel to those that hold for the modality S4. We know that modality in S4 doesn't collapse, i.e. $\Box\varphi$ is not always equivalent to φ. So we know that these principles don't entail that $\top\varphi$ is always equivalent to φ.

1	$\delta \leftrightarrow (\top\delta \rightarrow \varphi)$	Given, by definition(!?)
2	$\delta \rightarrow (\top\delta \rightarrow \varphi)$	From biconditional 1
3	$\top(\delta \rightarrow (\top\delta \rightarrow \varphi))$	By the rule of proof T3 from 2
4	$\top\delta \rightarrow \top(\top\delta \rightarrow \varphi)$	Using an instance of T1 and then Modus Ponens from 3
5	$\top\delta \rightarrow \top\top\delta$	By T2
6	$\top\delta$	Supposition
7	$\top\top\delta$	Modus Ponens from 5, 6
8	$\top(\top\delta \rightarrow \varphi)$	Modus Ponens from 4, 6
9	$\top\top\delta \rightarrow \top\varphi$	Using an instance of T1 and then Modus Ponens from 8
10	$\top\varphi$	Modus Ponens from 7, 9
11	$\top\varphi \rightarrow \varphi$	By T0
12	φ	Modus Ponens from 10, 11
13	$\top\delta \rightarrow \varphi$	Conditional Proof from subproof 6 to 12
14	$(\top\delta \rightarrow \varphi) \rightarrow \delta$	From biconditional 1
15	δ	Modus Ponens from 13, 14
16	$\top\delta$	by T3, since we've proved 15
17	φ	Modus Ponens from 13, 16

The conditional rules used here are the standard ones, so we again get a slightly sharpened take on Curry's paradox and a slightly sharpened theorem. The paradox is that it seems that we can form self-referential sentences, and our ordinary truth-predicate should satisfy (informal versions of) $\top 0$ to $\top 3$. The improved version of Theorem 70 is

Theorem 71. *If T is consistent, has a standard conditional and knows some arithmetic, it can't also have a truth-predicate T for which conditions $\top 0$ to $\top 3$ hold.*

Proof. Assume for reductio that T has a truth-predicate for which conditions $\top 0$ to $\top 3$ hold. Take any φ and form the predicate $\mathsf{T}(x) \rightarrow \varphi$. Since T knows

some arithmetic, so proves the Diagonalization Lemma, there is some δ such that $T \vdash \delta \leftrightarrow (\mathsf{T}(\overline{\ulcorner \delta \urcorner}) \rightarrow \delta)$, i.e. in shorthand $T \vdash \delta \leftrightarrow (\top\delta \rightarrow \delta)$. The proof then continues as above to derive φ. Since φ was arbitrary, that makes T inconsistent, contrary to hypothesis. □

Now, you've probably noticed that conditions $\top 1$ to $\top 3$ on the 'modality' $\top$ (defined from the truth-predicate $\mathsf{T}(x)$) exactly match, numbering apart, the so-called derivability conditions on $\Box$ (which we defined from the provability-predicate $\mathsf{Prov}(x)$). Here are those conditions again:

Defn. 62. *The derivability conditions hold in T if and only if, for any T-sentences φ, ψ,*

C1. If $T \vdash \varphi$, then $T \vdash \Box\varphi$,

C2. $T \vdash \Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$,

C3. $T \vdash \Box\varphi \rightarrow \Box\Box\varphi$.

So again let φ be any sentence, and consider the wff $(\mathsf{Prov}(\mathsf{z}) \rightarrow \varphi)$. Assuming T can prove the general Diagonalization Lemma (Theorem 47), we'll have the particular case that there is a δ such that T can prove $\delta \leftrightarrow (\mathsf{Prov}(\overline{\ulcorner \delta \urcorner}) \rightarrow \varphi)$, or for short, $T \vdash \delta \leftrightarrow (\Box\delta \rightarrow \varphi)$. Then, in the case where T also proves $\Box\varphi \rightarrow \varphi$, so we can replicate line 12 of the proof, the proof above goes through again, just with '$\Box$' substituted for $\top$.

Which proves **Löb's Theorem**:

Theorem 72. *If T knows some arithmetic, and the derivability conditions hold, then if $T \vdash \Box\varphi \rightarrow \varphi$ then $T \vdash \varphi$.*

This is a bit of a surprise: we might have expected that a theory which has a well-constructed provability-predicate should 'think' that if it can prove φ then indeed φ, i.e. we might have expected that in general $T \vdash \Box\varphi \rightarrow \varphi$. But not so. A respectable theory T can only prove this if in fact it can *already* show that φ.

By the way, this answer's **Henkin's question**. We know from the Diagonalization Lemma applied to the unnegated wff $\mathsf{Prov}(\mathsf{x})$ thatthere 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 (compare the Gödel sentence that 'says' it is unprovable). Henkin asked: *is* H provable? Well, we now know that 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.

12.4 The Second Incompleteness Theorem again

The argument from Löb's Theorem to Gödel's Second Theorem is indeed swift! Assume the conditions for Löb's Theorem apply. Then as a special case we get that if $T \vdash \Box\bot \to \bot$ then $T \vdash \bot$. Hence, if $T \nvdash \bot$, so T is consistent, then $T \nvdash \Box\bot \to \bot$, hence $T \nvdash \neg\Box\bot$ hence (just by definition of Con), $T \nvdash \mathsf{Con}$.

There's a converse too, i.e. the Second Theorem quickly entails Löb's Theorem (*IGT*2, §34.5). So the two theorems come to much the same.

So the situation is this. What Gödel saw in proving the first theorem, in the broadest terms, is that when we move from talking about truth to talking about provability, thinking about Liar-style sentences can lead not to paradox but to the First Incompleteness Theorem. Much later, Löb spotted that, similarly, when we move from talking about truth to talking about provability, Curry-paradox style reasoning again leads not to paradox but to the Second Theorem.

What's still to come ... And here, these notes will end (as did the lectures they were originally written to accompany). But *IGT2* carries on. So let me briefly indicate where the book now goes, and how it links in with what we've done so far.

OK: 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 we have worked Gödelian wonders with these very limited ingredients. But 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 any *general* theory of computable functions or (equivalently) on a general theory of effectively decidable properties and relations. (Recall: the p.r. functions are not all the computable functions, by Theorem 21.) Compare, then, our informal early theorems

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

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

A 'sufficiently strong theory' is, recall, one which can capture at least all effectively decidable numerical properties. So both those informal theorems *do* deploy the informal notion of effective decidability. And to prove an analogue of the undecidability theorem and to get the early incompleteness theorem to fit together nicely with our later official Gödelian proof, we'll therefore need to give a proper formal treatment of decidability.

And that shapes the main tasks in the remaining chapters of *IGT*2. In more detail

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 indeed comprise *all* total numerical functions which are effectively computable. (So the suggestion is that we can trade in the informal notion of *effective decidability* for the formally defined notion of *recursive decidability*.)

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 in this sense 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 incompleteness theorem, Theorem 5.

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 of the pivotal technical result that the Turing-computable (total) functions are in fact just the μ-recursive functions again. So Church's Thesis and Turing's Thesis come to the same.

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*: that is to say, 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.

5. We show that the unsolvability of the halting problem gives us a nice proof that it isn't mechanically decidable what's a theorem of first-order logic, and it also entails Gödelian incompleteness again.

6. 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.