

Gödel Without (Too Many) Tears – 1

Incompleteness – the very idea

Peter Smith

University of Cambridge

February 7, 2011

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

- The notion of effective decidability
 - What's a formalized language?
 - What's a formal axiomatized theory?
 - What's negation incompleteness?
 - 'Deductivism' about basic arithmetic
 - Two versions of Gödel's First Incompleteness Theorem
 - The First Theorem as an *incompleteness* theorem
 - How did Gödel prove (one version of) the First Incompleteness Theorem?
-

Why these notes? After all, I've already written a pretty detailed book, *An Introduction to Gödel's Theorems* (CUP, heavily corrected fourth printing 2009; henceforth *IGT*). 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 pretty 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 by mathematicians wanting a brisk introduction. You might reasonably want to get your heads around only those technical basics which are actually necessary for understanding how the theorems are proved and for appreciating philosophical discussions about incompleteness.

So you really need a cut-down version of the book – an introduction to the *Introduction!* Well, isn't that what lectures are for? Indeed. But now there's another snag. I haven't got many lectures to play with. So either (A) I crack on at a very fast pace (hard-core mathmo style), cover those basics, but perhaps leave too many people puzzled and alarmed. Or (B) I do relaxed talk'n'chalk,¹ highlighting the Really Big Ideas, making sure everyone is grasping them as we go along, but inevitably omit important stuff and leave quite a gap between what happens in the lectures and what happens in the book. What to do?

I'm going for plan (B). But then you will probably need something to help bridge that gap between lectures and book. Hence these notes.

The idea, then, is to give relaxed and very informal lectures, highlighting Really Big Ideas, not worrying too much about depth or fine-detail (nor even worrying about getting through *all* of the day's intended menu of topics). These notes then expand things just enough, and give pointers to relevant chunks of *IGT*. I hope these notes will to a fair extent be stand-alone, and tell a briefer but still coherent story read by themselves: so occasionally I'll copy a paragraph or two from the book, rather than just refer to them.

¹'Talk and white-board marker' doesn't have quite the same ring, does it?

But these notes do come with a Logical Health Warning: in the interests of relative brevity, I'll occasionally have to apply that good maxim 'Where it doesn't itch, don't scratch'. In other words, sometimes I'll say things that are not in their ideal official form, but I hope in unworrying ways that can be easily remedied if you are feeling pernickety.

The bullet-pointed headers to each helping of notes – to each episode, as I'll call it – are there to give pointers/reminders to the coverage.

A final introductory remark. If you notice any typos/thinkos in these notes and/or the latest printing of *IGT* please let me know (peter_smith@me.com). In due course, there will probably be a more substantially revised second edition of the book; so I'd also be very grateful for any more general comments about the book that might help me improve the book. Some further relevant materials, plus the very latest versions of these notes, can be found at www.logicmatters.net.

1 Kurt Gödel (1906–1978)

The greatest logician of the twentieth century. Born in what is now Brno. Educated in Vienna. Gödel left Austria for the USA in 1938, and spent rest of his life at the Institute of Advanced Studies at Princeton. Always a perfectionist, after the mid 1940s he more or less stopped publishing.

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 and seminal work on set theory, as well as make contributions to proof theory and to the philosophy of mathematics. He even wrote on models of General Relativity. Talk of 'Gödel's Theorems', however, typically refers to his two *incompleteness* theorems in an epoch-making 1931 paper.²

For a brief overview of his 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>. But there's a very nice full biography, John Dawson *Logical Dilemmas* (A. K. Peters, 1997), which is well worth reading as it will also give you a real sense of the logical scene in the glory days of the 1930s.

2 'On formally undecidable propositions of *Principia Mathematica* and related systems I'

This is the title of the 1931 paper which proves the First Incompleteness Theorem and states the Second Theorem. (The 'I' 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. But that was never written. I'll explain later why Gödel didn't need to bother.)

Even the title gives us a number of things to explain. What's a 'formally undecidable proposition'? What's *Principia Mathematica*? – you've heard of it, no doubt, but what's the project of that triple-decker work? What counts as a 'related system'? In fact, just what is meant by 'system' here? We'll take the last question first.

2.1 'Systems' – i.e. formal axiomatized theories

Our concern is with systems in the sense of *formal axiomatized theories*. T is such a theory if 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.

To explain, we first need a definition:

Defn. 1. A property P defined over a domain D is effectively decidable iff there's an algorithm (a finite set of instructions for a deterministic routine computation) for settling in a finite number of steps, for any $o \in D$, whether o has property P – i.e. there's a step-by-step mechanical routine

²Yes, Gödel proved a 'completeness theorem' and 'incompleteness theorems'. By the end of this first episode you should be able to tell the difference!

for settling the issue, a suitably programmed deterministic computer could in principle do the trick. A set Σ is effectively decidable if the property of being a member of that set is effectively decidable.

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. But there's an issue about terminology we'd better clarify.

A language, for us, is a bunch of symbols with an 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 unusual for logic-books to talk of a language when speaking just of uninterpreted strings of symbols. But I rather 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. For an effectively formalized language L , the basic alphabet of L is to be finite, and 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.

NB, the restriction to a finite basic vocabulary still allows us, e.g., to have an infinite supply of variables: for example, given the two symbols 'x' and '' we can construct an infinite supply of composite symbols x, x', x'', x''' .³ As to the effective decidability of the properties of being a term, etc., the point of setting up a formal language is usually (inter alia) precisely to put issues of what is and isn't a sentence beyond dispute, so we want to be able effectively to decide whether a string of symbols is or is not a sentence. A formal interpreted 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.

(ii) A theory T built in a formalized language L will have a certain class of L -sentences picked out as *axioms*.⁴ Again it is to be *effectively decidable* what's an axiom. (After all, if we are making a theory rigorous, but then can't routinely tell whether a given sentence is one of its axioms, that would – usually – be pretty pointless.)

(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 comes equipped with a proof-system, a set of rules for deducing further theorems from our initial axioms. But describing a proof-system such that we couldn't then routinely tell whether its rules are in fact being followed wouldn't have much point. Hence we naturally 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 (ii) and (iii),

Defn. 3. A formal axiomatized theory T has an effectively formalized language L , a certain class of L -sentences picked out as axioms where it is 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.

I say 'array of wffs' rather than simply 'sequence of wffs', because you might have come across proof-systems where deductions are arranged in tree-form rather than in a linear one-proposition-after-another form. The key thing is that, however proofs are laid out, we must be able to

³In some contexts, for technical purposes, there is interest in talking about formal languages with an uncountably infinite number of primitives. That's why the finiteness constraint needs to be made explicit.

⁴We'll allow the class of axioms to be 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!

mechanically work out whether an alleged proof does indeed conform to rules of T 's proof system. As a reminder, recall we allow the limiting case of writing down an axiom φ by itself to count as a one-line 'proof' with premiss φ and conclusion φ .

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 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 general way of telling of an arbitrary φ whether it is provable in T or not.)

2.2 Notational conventions

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 *IGT*, and indeed is not an uncommon one):

1. Particular expressions from formal systems – and abbreviations of them – will be in sans serif type. Examples: $\mathbb{S}0 + \mathbb{S}0 = \mathbb{S}\mathbb{S}0$, $\forall x \mathbb{S}x \neq 0$. [Blackboard convention: overline formal wffs when clarity demands. Bracketing will tend to be casual.]
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, as in the ' φ ' we've just used, are schematic variables in the metalanguage in which we talk about our formal systems.

For more explanations, see *IGT*, §§2.2, 3.1–3.3, 4.1.

2.3 'Formally undecidable propositions' and negation incompleteness

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 .

So NB, ' \vdash ' officially signifies provability in T , a formal syntactically definable relation, not semantic entailment.

Defn. 5. 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. 6. A sentence φ is formally undecidable by T iff $T \not\vdash \varphi$ and $T \not\vdash \neg\varphi$.

Another bit of terminology:

Defn. 7. 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 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. We fix on an interpreted formal language L apt for expressing such truths. And then we'd ideally like to build a theory T in L , whose axioms are 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 like T to be such that either $T \vdash \varphi$ or $T \vdash \neg\varphi$. Negation completeness, then, is a natural desideratum for theories.

For more explanations, see *IGT*, §3.4.

2.4 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*⁵ 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, i.e. for 0, *S0*, *SS0*, *SSS0*, ... – 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 the the 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 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 intuition is very puzzling and problematic. So we might well be tempted instead by Frege's view that the axioms are *analytic*, truths of logic-plus-definitions. On this view, we don't need Kantian 'intuitions' going beyond logic: logical reasoning from definitions is enough. The Fregean brand of deductivism is standardly dubbed 'logicism'.

Famously, Frege's attempt to be a logicist deductivist about arithmetic (in fact, for him, 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 [yep! – *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 big promisory 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 make good on that promise. The project is to set down some logical axioms and definitions and deduce the laws of basic arithmetic (and then more) from them. Famously, they 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'.

2.5 Gödel's bomb

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*.⁶ But leave those worries aside – they pale into insignificance compared with the bomb exploded by Gödel.

⁵The interpreted language, of course – or no questions are posed!

⁶*Principia* without the dodgy Axiom is a '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: for more on this see <http://plato.stanford.edu/entries/principia-mathematica/>

For Gödel’s First Incompleteness Theorem shows that any form of deductivism about even just basic arithmetic (not just *Principia*’s) is in 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!⁷

3 The First Incompleteness Theorem, a bit more carefully

3.1 Two versions of the First Theorem

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

Defn. 8. *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 as primitives or introduced by definition – and has a predicate whose extension is the natural numbers.*

The point of that last clause is that if ‘ N ’ is a predicate satisfied just by numbers, then the wff $\forall x(Nx \rightarrow \varphi(x))$ 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 ‘ $x = x$ ’ for ‘ N ’, or – equivalently – just forget about it!)

Defn. 9. *A theory T is sound iff its axioms are true (on the interpretation built in to T ’s language), and its logic is truth-preserving, so all its theorems are true.*

Defn. 10. *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, 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 $T \not\vdash G_T$ and $T \not\vdash \neg G_T$, 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 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. There were other reasons too for wanting to steer away from semantic notions, reasons to do with ‘Hilbert’s program’ about which more anon. 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:

⁷‘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.

Theorem 2. *For any consistent formal axiomatized theory T which can prove a certain modest amount of arithmetic (and has a certain additional desirable property that any sensible formalized arithmetic will share), there is a sentence of basic arithmetic G_T such that $T \not\vdash G_T$ and $T \not\vdash \neg G_T$, 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. The 'contains 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 that 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 – how we can drop that condition.)

For the present, however, let's concentrate on the semantic version of Gödel's theorem, i.e. Theorem 1.

3.2 Theorem 1 is better called an *incompleteness* theorem

Suppose T is a sound theory which can express claims of basic arithmetic. Then we can find a true G_T such that $T \not\vdash G_T$ and $T \not\vdash \neg G_T$. *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 .*

Now, we might want to 'repair the gap' in T by adding G_T as a new axiom. So consider the theory $U = T + G_T$ (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 an specified axiom to T doesn't make it undecidable what is an axiom of the augmented theory. (iii) U still can express claims of basic arithmetic. So Gödel's First Incompleteness Theorem applies, and we can find a sentence G_U such that $U \not\vdash G_U$ and $U \not\vdash \neg G_U$. And since U is stronger than T , we have a fortiori, $T \not\vdash G_U$ and $T \not\vdash \neg G_U$. 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*.

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

Let's take a first pass at outlining how Gödel proved the semantic version of his incompleteness theorem. Obviously we'll be coming back to this in a lot more detail later, but we can give just a flavour of what's going on. We kick off with two natural definitions.

Defn. 11. *If L contains the language of basic arithmetic, so it contains a term 0 for zero and a function expression S for the successor function, then the terms $0, S0, SS0, SSS0, \dots$, are L 's standard numerals, and we'll use ' \bar{n} ' to abbreviate the standard numeral for n .⁸*

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

Defn. 12. *The formal wff $\varphi(x)$ of the interpreted language L expresses the numerical property P iff $\varphi(\bar{n})$ is true on interpretation just when n has property P . Similarly, the formal wff $\psi(x, y)$ expresses the numerical relation R iff $\psi(\bar{m}, \bar{n})$ is true just when m has relation R to n . And the formal wff $\chi(x, y)$ expresses the numerical function f iff $\chi(\bar{m}, \bar{n})$ is true just when $f(m) = n$.*

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

Then the proof of Theorem 1 in outline form goes as follows:

⁸So what is the relation between 0 and $\bar{0}$?

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 a formal axiomatized theory T , we can define e.g. the numerical properties Wff_T , $Sent_T$ and Prf_T where

$Wff_T(n)$ iff n is the code number of a T -wff.

$Sent_T(n)$ iff n is the code number of a T -sentence.

$Prf_T(m, n)$ iff m is the code number of a T -proof of the T -sentence with code number n .

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 [takes a bit of work!]. 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 $Prf_T(x, y)$ that expresses the property Prf_T , so $Prf_T(\bar{m}, \bar{n})$ 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 is that the language of basic arithmetic is good at expressing decidable properties and relations, and relations like $Prf_T(m, n)$ are decidable because T is a formalized theory.

3. *The construction: building a Gödel sentence* Now put $Prov_T(\bar{n}) =_{\text{def}} \exists x Prf_T(x, \bar{n})$. This says that the wff with number n is a theorem.

Next – the really cunning bit, but surprisingly easy – we show how to build a ‘Gödel’ sentence G_T such that G_T is in fact equivalent to $\neg Prov_T(\bar{g})$, where the standard numeral ‘ \bar{g} ’ is the numeral denoting the code-number for G_T . In other words (think about it!!), G_T is true if and only if G_T isn’t a theorem.

4. *The argument* Suppose $T \vdash G_T$. 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 hypothesis. So $T \not\vdash G_T$. So G_T is true. So $\neg G_T$ is false and T , being sound, can’t prove it. Hence we also have $T \not\vdash \neg G_T$.

There are big gaps to fill there, but that’s the overall strategy. (The proof of Theorem 2 then shows that we can get the same result using the same construction of a Gödel sentence by dropping the assumption that T is sound, so long as we require a bit more by way of what the theory T can prove, and require T to have that currently mysterious ‘additional desirable 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 sentences that say ‘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. I hope that over the coming episodes, the answer to that good question will become clear!

Now read *IGT*, §§1.1–3.4. (“But hold on! What’s the Second Theorem that you mentioned?” Good question – the opening chapter of *IGT* will tell you: but here in these notes we’ll maintain the suspense as far as Episode 10!)

Gödel Without (Too Many) Tears – 2

Incompleteness and undecidability

Peter Smith

University of Cambridge

January 16, 2011

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

- The idea of a decidable theory
 - Thm. 3: Any consistent, negation-complete, axiomatized formal theory is decidable
 - Expressing and capturing properties, relations and functions
 - 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
 - Two corollaries, about the undecidability of first-order logic, and the unaxiomatizability of True Basic Arithmetic
-

In Episode 1, we introduced the very idea of a negation-incomplete formalized theory T . We noted that if we are aiming to construct a theory of basic arithmetic, we’ll 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, 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 different sort of proof of incompleteness, one that doesn’t go via any explicitly self-referential construction. This proof will also 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 . . .

5 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. 13. 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 will 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 T -theorems (any T -theorem has a proof, and by churning through all possible strings of symbols, we eventually churn through to reach any possible 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, which is guaranteed to deliver a verdict in a finite number of steps. \square

We are, of course, relying here on a very, *very*, relaxed notion of effective decidability-in-principle in Defn. 1, 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 *IGT*, §2.2).

6 Capturing numerical properties in a theory

Here's an equivalent way of rewriting the earlier Defn. 12:

Defn. 14. A property P is expressed by the open wff $\varphi(x)$ 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 $\varphi(\bar{n})$ is true,
- ii. if n does not have the property P , then $\neg\varphi(\bar{n})$ is true.

A two-place relation R is expressed by the open wff $\psi(x, y)$ with two free variables iff, for every m, n ,

- i. if m is R to n , then $\psi(\bar{m}, \bar{n})$ is true,
- ii. if m is not R to n , then $\neg\psi(\bar{m}, \bar{n})$ is true.

A one-place function f is expressed by the open wff $\chi(x, y)$ with two free variables iff, for every m, n ,

- i. if $f(m) = n$, then $\chi(\bar{m}, \bar{n})$ is true,
- ii. if $f(m) \neq n$, then $\neg\chi(\bar{m}, \bar{n})$ is true.

(Recall, \bar{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. 15. *The theory T captures the property P by the open wff $\varphi(x)$ iff, for any n ,*

- i. if n has the property P , then $T \vdash \varphi(\bar{n})$,*
- ii. if n does not have the property P , then $T \vdash \neg\varphi(\bar{n})$.*

The theory T captures the two-place relation R by the open wff $\psi(x, y)$ iff, for any m, n ,

- i. if m is R to n , then $T \vdash \psi(\bar{m}, \bar{n})$,*
- ii. if m is not R to n , then $T \vdash \neg\psi(\bar{m}, \bar{n})$*

The theory T captures the one-place function f by the open wff $\chi(x, y)$ iff, for any m, n ,

- i. if $f(m) = n$, then $T \vdash \chi(\bar{m}, \bar{n})$,*
- ii. if $f(m) \neq n$, then $T \vdash \neg\chi(\bar{m}, \bar{n})$.*

So: what a theory can *express* depends on the richness of its language; what a theory can *capture* (mnemonic: case-by-case prove⁹) depends on the richness of its axioms and rules of inferences.

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 $\varphi(x)$ which expresses the decidable property P , and (b) is such that if n has property P , $T \vdash \varphi(\bar{n})$, and if n does not have property P , $T \vdash \neg\varphi(\bar{n})$. 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. 16. *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.

7 Sufficiently strong theories are undecidable

We now prove a lovely theorem:

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. 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

⁹Many say 'represent' instead of my 'capture'.

$\varphi_0(x), \varphi_1(x), \varphi_2(x), \varphi_3(x), \dots$

For we can just churn out all the strings of symbols of T 's language, 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 $T \vdash \neg\varphi_n(\bar{n})$.

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.¹⁰

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, $\varphi_n(x)$. Then it is a mechanical matter to form the numeral \bar{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 $\neg\varphi_n(\bar{n})$ 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 $\varphi(x)$. Let's suppose the d -th wff does the trick: that is to say, property D is captured by $\varphi_d(x)$.

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

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

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

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

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 $T \vdash \neg\varphi_d(\bar{d})$.

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

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

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 a theory T , and then just pass T over to a computer to tell us what's a theorem and what isn't.

8 'Diagonalization'

It is worth remarking on the key construction here. We take a sequence of wffs $\varphi_n(x)$ (for $n = 0, 1, 2, \dots$) and then considering the (negations of) the wffs $\varphi_0(\bar{0})$, $\varphi_1(\bar{1})$, $\varphi_2(\bar{2})$, etc.¹¹ This sort of thing is called a *diagonalizing*. Why?

¹⁰And 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.

¹¹Reality check: what is the relation between $\varphi(\bar{0})$ and $\varphi(0)$?

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

9 Incompleteness again

Theorem 3 says: any consistent, negation-complete, axiomatized formal theory is decidable. Theorem 4 says: 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 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. So there is a major promisory note implicit here: we can 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. And indeed the trick can be done. However, supplying and defending the sharp account of the notion of decidability 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 proving incompleteness by Gödel's original method. So over the next episodes, we are going to revert to exploring Gödel's route to the incompleteness theorems.

10 A corollary about the decidability of logic

To finish, it is worth noting two corollaries of our results so far. But before reading on, do double-check that you fully understand the three theorems in this episode so far.

Happy? Then first, here's a corollary from our undecidability result. We start an obvious definition:

Defn. 17. *A formalized logic is decidable iff the property of being a theorem of the logic – i.e. a sentence deducible from no premisses – is decidable.*

It is familiar that standard propositional logic is decidable (doing a truth-table test or a tree test decides what's a tautology, and the theorems are all and only the tautologies). It is familiar too that there's no obvious analogue to the truth-table test for deciding of an arbitrary sentence whether it is theorem of standard first-order logic (a.k.a. the predicate calculus). But is there some other decision procedure?

Well, Theorem 4 quickly implies:

Theorem 6. *If there is a consistent theory with a first-order logic which is sufficiently strong and has a finite number of axioms, then first-order logic is undecidable.*

Proof Suppose Q is a consistent finitely axiomatized theory with a first-order logic and which is sufficiently strong. Since it is finitely axiomatized, we can wrap all its axioms together into one long conjunction, \hat{Q} . And then, trivially, $Q \vdash \varphi$ if and only if $\vdash \hat{Q} \rightarrow \varphi$; i.e. we can prove φ inside Q if and only if a certain related conditional is logically provable from no assumptions. So if (1) the logic were decidable, then (2) we could mechanically tell whether the conditional $\hat{Q} \rightarrow \varphi$ is a logical theorem, hence (3) we could mechanically decide whether φ is a Q -theorem. But since Q is a consistent sufficiently strong formalized theory (3) is impossible. So (1) is impossible – the logic must be undecidable. \square

Much later, we'll find that there is indeed a consistent, finitely axiomatized, weak arithmetic with a first-order logic, which is sufficiently strong – the so-called Robinson Arithmetic \mathcal{Q} fits the bill. So that will settle it: first-order logic really is undecidable.

11 A corollary about axiomatizability

Again we start with definitions:

Defn. 18. *A set of sentences Σ can be formally axiomatized iff there is an axiomatized formal theory T such that Σ is the set of T -theorems.*

Defn. 19. *An interpreted language L is sufficiently expressive iff, for every decidable property P of numbers, there is a predicate $\varphi(x)$ which expresses P .*

Then we immediately have

Theorem 7. *The set of truths of a sufficiently expressive language L is not formally axiomatizable by a theory framed in L .*

Proof Suppose, for reductio, that T is (i) a formalized theory in language L which proves all the truths expressible in L . T will (ii) then be negation complete (since by hypothesis it will prove the true one out of any $\varphi/\neg\varphi$ pair). But also T will (iii) be sufficiently strong: for by hypothesis, if P is a decidable property, T can express it using some $\varphi(x)$, and it proves all truths so if $\varphi(\bar{n})$ is true (so n is P) then $T \vdash \varphi(\bar{n})$, and if $\neg\varphi(\bar{n})$ is true (so n is not P) then $T \vdash \neg\varphi(\bar{n})$.

But the combination of (i), (ii) and (iii) contradicts our incompleteness theorem: so there can't be such a theory as T . □

Later we will see that even L_A , the language of basic arithmetic is sufficiently expressive: so it will follow that we can't axiomatize True Basic Arithmetic, i.e. the set of L_A -truths, by any theory couched in L_A . (In fact, we can improve on that result.)

At this point, you can usefully read Chs 4 and 6 of *IGT*. (You might also look at Ch. 5 – but proof details there are perhaps only for enthusiasts who are more mathematically minded. In fact the arguments are about as tricky as any in the book, and I now wonder whether I should have included them. I certainly don't want you to get fazed by them!)

Afterword for enthusiasts By all means skip this extended footnote. I hesitate to include it because I don't want anyone to get bogged down at this early stage of the notes! However, the point isn't without interest.

Someone, such as an earlier time-slice of me, might be tempted by the following further thought in response to Theorem 5:

Suppose we start off with a consistent 'sufficiently strong' theory T couched in some language which just talks about arithmetic matters: then this theory T is incomplete, and will have arithmetical formally undecidable sentences. But now imagine that we extend T 's language (perhaps it now talks about sets of numbers as well as about numbers), and we give it richer axioms, to arrive at an expanded consistent but still formally axiomatized theory U . U will still be sufficiently strong if T is, and so Theorem 5 will still apply given that U is properly axiomatized. Note, however, that as far as Theorem 5 is concerned, it could be that U repairs the gaps in T and proves every truth statable in T 's language, while the incompleteness has now 'moved outwards', so to speak, to claims involving U 's new vocabulary.

If that were right, this would indicate another way Theorem 5 is weaker than Gödel's canonical result, for *he* shows that some incompleteness will always remain even in a theory's arithmetical core, even as it gets augmented with more and more additional axioms (so long as the resulting bigger theory remains properly axiomatized).

But as David Makinson pointed out to me, that displayed line of thought isn't right. It is worth pausing to see why so.

Consider the theory T^* whose axioms are just all the theorems of U statable in the original narrower language of T , and which has no rules of inference. Now, what do we know about this theory? Well, we don't yet know that it counts as a kosher formalized theory, because we don't know that it is decidable axiomatized. But we do know

1. T^* is consistent. T^* proves just some of U 's theorems, and U is consistent.
2. T^* is sufficiently strong. Because T^* includes all T 's theorems and T is already sufficiently strong.
3. If U proves every truth statable in T 's language, then T^* is negation complete. That's immediate from the definitions.

We now show that T^* isn't negation complete, and hence (by 3) U can't prove every truth statable in T 's language, contrary to the displayed line of thought above.

Proceed by reductio. Suppose T^* is negation complete. Then T^* must be decidable (by the same kind of argument used in proving Theorem 3). To spell that out: given any sentence A in the language of T^* , i.e. the language of T , just mechanically grind out theorems of U until you get either A or its negation (you must get one of them, since by supposition T^* is negation complete and by definition T^* 's theorems are just U 's theorems in the language of T , so U is negation complete in the language of T). If A is a U -theorem, then it is a T^* axiom; if its negation is a U -theorem, then A is not a T^* axiom. Since, given our assumption, it is decidable what is a T^* axiom, T^* would count as a kosher formalized theory by Defn. 3. So T^* would be a consistent, sufficiently strong, axiomatized formal theory. So by Theorem 5, T^* can't be negation complete after all, contrary to hypothesis. Q.E.D.

Two weak arithmetics

Peter Smith

University of Cambridge

November 3, 2010

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

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

Before continuing any further, a very general word about the contents of these handouts (and indeed of this course). As you will have seen, there is relatively little *philosophy* so far. We said, in very general terms, that the Gödelian incompleteness theorem sinks varieties of what I called “deductivism” as a framework for a philosophical position about basic arithmetic. But otherwise, our discussions up to this point have mostly been *mathematical* (albeit rather informal maths for the most part).

And that’s the way things will continue. As I see it, what will most help get you into a position to understand and assess various philosophical claims about Gödel’s results is to have a really solid grasp of what his theorems actually say and how they are proved. So these handouts will remain primarily an introduction to the *mathematics* of incompleteness. The *philosophical* implications are for you to draw out.

Back to work then. But so far, we’ve been going at a pretty rapid pace (just to get to exciting stuff fast!). We now need to slow right down.

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. Later, we announced that there actually *is* a consistent (and finitely axiomatized) weak arithmetic with a first-order logic 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 introduces a couple of weak arithmetics. Frankly, this will be pretty unexciting stuff – by all means skip fairly lightly over some of the more 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.

12 Baby Arithmetic

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

Question: 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 $T \not\vdash G_T$ and $T \not\vdash \neg G_T$, so T is negation incomplete.*

Answer: BA's very limited language L_B lacks quantifiers, so doesn't contain the language of basic arithmetic (see again Defn. 8). Because its language is so limited, the theory can in fact prove or disprove every sentence constructible in L_B .

12.1 The language L_B

The language L_B has non-logical symbols

$0, S, +, \times$

The first of these is of course a constant (intended to denote 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 ' \times ' as 'infix' function symbols in the usual way – i.e. we write $S0 + SS0$ rather than prefix the function sign as in $+S0SS0$ – then we'll also need brackets for scoping the function signs, to disambiguate $S0 + SS0 \times SSS0$, e.g. as $(S0 + (SS0 \times SSS0))$.

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', '+', ' \times '. So, examples are 0, SSS0, (S0 + SS0), ((S0 + SS0) \times 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 ' \bar{n} ' to represent the standard numeral $SS \dots S0$ with n occurrences of 'S'. Thus ' $\bar{3}$ ' is short for 'SSS0'. The value of ' \bar{n} ' is of course n .¹²

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, $(S0 + SS0) = SSS0$ is a true atomic wff – which we can abbreviate, dropping brackets in a natural way, as $\bar{1} + \bar{2} = \bar{3}$ – and $(S0 \times SS0) = SSS0$ 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 *IGT*, 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!]

12.2 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.

¹²Recall also our convention from §2.2: **sans serif** expressions belong to L_B , or whatever formal language is under discussion: *italic* symbols are just part of everyday mathematical English.

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 place-holders ‘ ζ ’, ‘ ξ ’ is an axiom.*

Schema 1. $0 \neq S\zeta$

Schema 2. $S\zeta = S\xi \rightarrow \zeta = \xi$

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 larger number Sn to m is governed by the rule: for any m, n , $m + Sn = S(m + n)$. Those two principle 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 .

Again, 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. $\zeta + 0 = \zeta$

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

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

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

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

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

To summarize, then,

Defn. 20. *BA is the theory whose language is L_B , logic is propositional logic plus standard identity rules, and whose non-logical axioms are every numerical instance of Schemas (1) to (6).*

12.3 Variables and variables

We better pause to comment on that use of metalinguistic variables like ‘ ξ ’. These are *not* part of the language L_B . They are added to informal mathematical English to expedite our talking about wffs in the language L_B . If you are really pernickety, you could write, for example,

for any numerals ζ, ξ , $\ulcorner S\zeta = S\xi \rightarrow \zeta = \xi \urcorner$ is an axiom,

where the corners are Quine quotes.

Ordinary mathematical English is of course casual about such matters, and sometimes recruits object-language variables x, y etc. to do the work of expressing schematic generalizations. That’s usually harmless, but we’d better be more careful, even if we don’t get ultra careful with Quine quotes.

12.4 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 $BA \vdash 4 \neq \bar{2}$, i.e. $BA \vdash SSSS0 \neq SS0$

| | | |
|----|-------------------------------------|------------------------------|
| 1. | $SSSS0 = SS0$ | Supposition |
| 2. | $SSSS0 = SS0 \rightarrow SSS0 = S0$ | Axiom, instance of Schema 2 |
| 3. | $SSSS0 = S0$ | From 1, 2 by MP |
| 4. | $SSSS0 = S0 \rightarrow SS0 = 0$ | Axiom, instance of Schema 2 |
| 5. | $SS0 = 0$ | From 3, 4 by MP |
| 6. | $0 \neq SS0$ | Axiom, instance of Schema 1 |
| 7. | Contradiction! | From 5, 6 and identity rules |
| 8. | $SSSS0 \neq SS0$ | From 1 to 7, by RAA. |

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

Theorem 8. *If s and t are distinct numbers, then $BA \vdash \bar{s} \neq \bar{t}$.*

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

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

| | | |
|----|--|-----------------------------|
| 1. | $SS0 \times 0 = 0$ | Axiom, instance of Schema 5 |
| 2. | $SS0 \times S0 = (SS0 \times 0) + SS0$ | Axiom, instance of Schema 6 |
| 3. | $SS0 \times S0 = 0 + SS0$ | From 1, 2 by LL |

(‘LL’ of course indicates the use of Leibniz’s Law to intersubstitute identicals.) To proceed, we now need to show that $0 + SS0 = SS0$. 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 Schema 3 |
| 5. | $0 + S0 = S(0 + 0)$ | Axiom, instance of Schema 4 |
| 6. | $0 + S0 = S0$ | From 4, 5 by LL |
| 7. | $0 + SS0 = S(0 + S0)$ | Axiom, instance of Schema 4 |
| 8. | $0 + SS0 = SS0$ | From 6, 7 by LL |

Which gives us what we want:

| | | |
|----|-----------------------|-----------------|
| 9. | $SS0 \times S0 = SS0$ | From 3, 8 by LL |
|----|-----------------------|-----------------|

That’s pretty laborious, 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 9. *BA can prove any true equation of the form $\bar{m} + \bar{n} = \bar{t}$ or $\bar{m} \times \bar{n} = \bar{t}$.*

In other words, BA can correctly add or multiply any two numbers. [Exercise for the mathematical: 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 10. *Suppose τ is a term of L_B and the value of τ on the intended interpretation of the symbols is t . Then $BA \vdash \tau = \bar{t}$.*

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

| | | |
|----|---|--------------------|
| 1. | $(\bar{2} + \bar{3}) \times (\bar{2} \times \bar{2}) = (\bar{2} + \bar{3}) \times (\bar{2} \times \bar{2})$ | Trivial! |
| 2. | $\bar{2} + \bar{3} = \bar{5}$ | Instance of Thm. 9 |
| 3. | $(\bar{2} + \bar{3}) \times (\bar{2} \times \bar{2}) = \bar{5} \times (\bar{2} \times \bar{2})$ | From 1, 2 using LL |
| 4. | $\bar{2} \times \bar{2} = \bar{4}$ | Instance of Thm. 9 |
| 5. | $(\bar{2} + \bar{3}) \times (\bar{2} \times \bar{2}) = \bar{5} \times \bar{4}$ | From 3, 4 using LL |
| 6. | $\bar{5} \times \bar{4} = \bar{20}$ | Instance of Thm. 9 |
| 7. | $(\bar{2} + \bar{3}) \times (\bar{2} \times \bar{2}) = \bar{20}$ | 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 be evaluated at each stage, eventually equating the complex formula with a standard numeral. [Exercise for the mathematical: give a proper argument ‘by induction on the complexity of the formula’ that proves Thm 10.]

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

Our little theorems now enable us to prove the following:

Theorem 11. *Suppose σ and τ are terms of L_B . Then if $\sigma = \tau$ is true, then $\text{BA} \vdash \sigma = \tau$, and if $\sigma = \tau$ is false, then $\text{BA} \vdash \sigma \neq \tau$.*

Proof Let σ evaluate to s and τ evaluate to t . Then, by Theorem 10, (i) $\text{BA} \vdash \sigma = \bar{s}$ and (ii) $\text{BA} \vdash \tau = \bar{t}$.

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

Suppose, on the other hand, that $\sigma = \tau$ is false, so $s \neq t$. Then by Theorem 8, $\text{BA} \vdash \bar{s} \neq \bar{t}$, and together with (i) and (ii) that implies $\text{BA} \vdash \sigma \neq \tau$, again by the logic of identity. \square

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

Theorem 12. *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 $\text{BA} \vdash \varphi$ or $\text{BA} \vdash \neg\varphi$. Hence BA is negation complete. \square

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, prescindng 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!

13 Robinson Arithmetic

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.

13.1 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, $S, +, \times$ expressing successor, addition and multiplication. But now we allow ourselves 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. 8. (For, of course, L_A has the the predicate expression ' $x = x$ ' which has the numbers as its extension, so fits our official definition, if we want to fuss about that.)

13.2 The axioms and logic of Q

The theory Q 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. $\forall x(0 \neq Sx)$

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

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 '0, S0, SS0, ...'. 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. $\forall x(x \neq 0 \rightarrow \exists y(x = Sy))$

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

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

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

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

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

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

Defn. 21. *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 in 1952 – i.e. long after Gödelian incompleteness was discovered.

13.3 Q 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 13. *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 $0 + \bar{n} = \bar{n}$. *But it can't prove the corresponding universal generalization:*

Theorem 14. $Q \not\vdash \forall x(0 + x = x)$.

Proof Since \mathbb{Q} is a theory with a standard first-order theory, for any L_A -sentence φ , $\mathbb{Q} \vdash \varphi$ if and only if $\mathbb{Q} \models \varphi$ (that's just the completeness theorem for first-order logic). Hence one way of showing that $\mathbb{Q} \not\vdash \varphi$ is to show that $\mathbb{Q} \not\models \varphi$: 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 \mathbb{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 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 \mathbb{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 \mathbb{Q} true and $\forall x(0 + x = x)$ false. So $\mathbb{Q} \not\vdash \forall x(0 + x = x)$ \square

Theorem 15. \mathbb{Q} is negation-incomplete.

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

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 \mathbb{Q} is incomplete. \mathbb{Q} is, so to speak, *boringly* incomplete.

13.4 \mathbb{Q} can capture *less-than-or-equals*

We’ve just seen something that \mathbb{Q} can’t do: now for something it *can* do. We’ll establish

Theorem 16. In \mathbb{Q} , the less-than-or-equal-to relation is captured by the wff $\exists v(v + x = y)$.

Given the definition of capturing, Defn 15, that means we need to show that, for any particular pair of numbers, m, n , if $m \leq n$, then $\mathbb{Q} \vdash \exists v(v + \bar{m} = \bar{n})$, and otherwise $\mathbb{Q} \vdash \neg\exists v(v + \bar{m} = \bar{n})$.

Proof Suppose $m \leq n$, so for some $k \geq 0$, $k + m = n$. \mathbb{Q} can prove everything BA proves and hence, in particular, can prove every true addition sum. So we have $\mathbb{Q} \vdash \bar{k} + \bar{m} = \bar{n}$. But

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

Suppose alternatively $m > n$. We need to show $Q \vdash \neg \exists v(v + \bar{m} = \bar{n})$. 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. | $\exists v(v + SS0 = S0)$ | Supposition |
| 2. | $a + SS0 = S0$ | Supposition |
| 3. | $a + SS0 = S(a + S0)$ | From Axiom 5 |
| 4. | $S(a + S0) = S0$ | From 2, 3 by LL |
| 5. | $a + S0 = S(a + 0)$ | From Axiom 5 |
| 6. | $SS(a + 0) = S0$ | From 4, 5 by LL |
| 7. | $a + 0 = a$ | From Axiom 4 |
| 8. | $SSa = S0$ | From 6, 7 by LL |
| 9. | $SSa = S0 \rightarrow Sa = 0$ | From Axiom 2 |
| 10. | $Sa = 0$ | From 8, 9 by MP |
| 11. | $0 = Sa$ | From 10 |
| 12. | $0 \neq Sa$ | From Axiom 1 |
| 13. | Contradiction! | From 11, 12 |
| 14. | Contradiction! | $\exists E$ 1, 2–13 |
| 15. | $\neg \exists v(v + SS0 = S0)$ | RAA 1–14 |

The only step to explain is at line (14) where we use a version of the Existential Elimination rule: if the temporary supposition $\varphi(a)$ leads to contradiction, for arbitrary a , then $\exists v\varphi(v)$ must lead to contradiction. And having done the proof for the case $m = 2, n = 1$, inspection reveals that we can use the same general pattern of argument to show $Q \vdash \neg \exists v(v + \bar{m} = \bar{n})$ whenever $m > n$. [Exercise: convince yourself that this claim is true!] So we are done. \square

13.5 Adding ' \leq ' to Q

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 we put for ' ξ ' and ' ζ ', $\xi \leq \zeta$ is just short for $\exists v(v + \xi = \zeta)$, 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 $\xi < \zeta =_{\text{def}} \exists v(Sv + \xi = \zeta)$.)

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.

13.6 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 17. *Q can capture all decidable numerical properties – i.e. it is sufficiently strong in the sense of Defn 16.*

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.

For parallel reading to this episode, see *IGT*, Ch. 8, and Ch. 9, §§9.1–9.4.

Gödel Without (Too Many) Tears – 4

First-order Peano Arithmetic

Peter Smith

University of Cambridge

November 5, 2010

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

The main course:

- 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

Dessert:

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

This episode, after the preamble, falls into two parts. First I introduce the canonical first-order theory of arithmetic, PA. Then – tacked on here, because I need to fit it in somewhere soon and this is as good a place as any – I introduce some terminology for distinguishing wffs on the basis of their ‘quantifier complexity’. *Do make sure you understand the idea of induction, and how that is handled in PA, before reading the rest of the episode.*

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 $T \not\vdash G_T$ and $T \not\vdash \neg G_T$, 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’re 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 property that any sensible formalized arithmetic will share), there is a sentence of basic arithmetic G_T such that $T \not\vdash G_T$ and $T \not\vdash \neg G_T$, 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 15. *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 $Q \not\vdash \forall x(0 + x = x)$. And since Q is sound on the built-in interpretation of its language, we also have $Q \not\vdash \neg\forall x(0 + x = x)$.

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. Also we have

Theorem 17. *Q can capture all decidable numerical properties – i.e. it is sufficiently strong in the sense of Defn 16.*

so a theory’s containing Q makes it a ‘sufficiently strong’ theory in the sense of Theorem 5. Of course establishing *these* facts is a non-trivial task for later: but they do explain why Q is so interesting despite its weakness.

Now read on

14 Arithmetical Induction

For a moment, put $\varphi(x)$ for $(0 + x = x)$. Then, as we noted, for any particular n , $Q \vdash \varphi(\bar{n})$, for Q can prove an unquantified true equation. But we showed that $Q \not\vdash \forall x\varphi(x)$. In other words, Q can separately prove all instances of $\varphi(\bar{n})$ but can’t prove the corresponding simple generalization. So let’s consider what proof-principle we might add to Q to fill this sort of yawning gap.

14.1 The ω -rule

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

$$\frac{\begin{array}{cccc} \vdots & \vdots & \vdots & \vdots \\ \varphi(\bar{0}) & \varphi(\bar{1}) & \varphi(\bar{2}) & \varphi(\bar{3}) & \vdots \end{array}}{\forall x\varphi(x)}$$

This rule – or rather the generalized version for any φ – is what’s called *the ω -rule*. It is evidently a sound rule: if each $\varphi(\bar{n})$ is true then indeed all numbers n satisfy $\varphi(x)$. 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 a formal axiomatized 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 in fact argue. The finiteness requirement we impose on formalized theories is, for that reason, not arbitrary. And so we’ll stick to that requirement, and hence have to ban the ω -rule.

14.2 Replacing an infinitary rule with a finite one

To repeat, as well as proving $\varphi(0)$, Q also proves $\varphi(\bar{1}), \varphi(\bar{2}), \varphi(\bar{3}), \dots$. And 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 that we have the following proof in Q:

| | | |
|----|---|----------------------------------|
| 1. | $\varphi(a)$ | Supposition |
| 2. | $0 + a = a$ | Unpacking the definition |
| 3. | $S(0 + a) = Sa$ | From 2 by LL |
| 4. | $(0 + Sa) = S(0 + a)$ | Instance of Axiom 5 |
| 5. | $(0 + Sa) = Sa$ | From 3, 4 |
| 6. | $\varphi(Sa)$ | Applying the definition |
| 7. | $(\varphi(a) \rightarrow \varphi(Sa))$ | From 1, 6 by Conditional Proof |
| 8. | $\forall x(\varphi(x) \rightarrow \varphi(Sx))$ | From 7, since a was arbitrary. |

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

In this way, $\varphi(0)$ and $\forall x(\varphi(x) \rightarrow \varphi(Sx))$ together prove each of of $\varphi(\bar{0}), \varphi(\bar{1}), \varphi(\bar{2}), \dots$, and those truths in turn, by the sound ω -rule, entail $\forall x\varphi(x)$. Thus putting everything together we have ...

$$\begin{array}{c}
 \varphi(0) \quad \frac{\varphi(0) \quad \frac{\forall x(\varphi(x) \rightarrow \varphi(Sx))}{\varphi(0) \rightarrow \varphi(S0)}}{\varphi(S0)} \quad \frac{\varphi(0) \quad \frac{\forall x(\varphi(x) \rightarrow \varphi(Sx))}{\varphi(0) \rightarrow \varphi(S0)}}{\varphi(S0)} \quad \frac{\varphi(S0) \quad \frac{\forall x(\varphi(x) \rightarrow \varphi(Sx))}{\varphi(S0) \rightarrow \varphi(SS0)}}{\varphi(SS0)} \quad \dots \dots \\
 \hline
 \forall x\varphi(x) \quad \omega\text{-rule}
 \end{array}$$

But now suppose we cut out all those (infinitely many) intermediate steps in that motivating argument! This act of radical compression will leave us a nice *finitary* rule:

Defn. 22. *The Induction Rule:*

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

This cut-down rule is evidently sound whatever predicate we put for $\varphi(\xi)$ which expresses a genuine numerical property. Add this finitary rule to Q and we’ll evidently at least patch the gap we found.

14.3 Induction: the basic idea

The basic idea reflected in that formal 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 *principle of arithmetical induction*, and is a standard method of proof for establishing arithmetical generalizations.

For those not so familiar with this standard method, here's a little example of the principle at work in an everyday informal mathematical context. Suppose we want to establish that the sum of the first n numbers is $n(n+1)/2$. Well, first define $\psi(n)$ so that

$$\psi(n) \leftrightarrow 1 + 2 + 3 \dots n = n(n+1)/2$$

Then (i) trivially $\psi(0)$ is true (the sum of the first zero numbers is zero)! And (ii) now suppose that $\psi(n)$ is true, i.e. $1 + 2 + 3 \dots n = n(n+1)/2$. Then the sum of the first $n+1$ numbers is $n(n+1)/2 + (n+1)$. Rearranging, that equals $(n+1)(n+2)/2$, i.e. $(Sn)(Sn+1)/2$. Which means that $\psi(Sn)$ will be true too. So (i) $\psi(0)$, and (generalizing) (ii) for all numbers n , if $\psi(n)$, then $\psi(Sn)$. Therefore, as we want, by induction the claim holds for all numbers.

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. Define $\chi(n)$ to be true if the conclusions of proofs appealing only to logical axioms 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 must either be an 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.

14.4 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. Number off the swans, for example, and let $\varphi(n)$ say that swan $\#n$ is white. We sample some swans and run (say) k checks showing that $\varphi(0), \varphi(1), \varphi(2), \dots, \varphi(k)$ are all true. We hope that these are enough to be representative of the whole population of swans, and so – taking a chance – 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. $\forall x(\varphi(x) \rightarrow \varphi(Sx))$. We put that universal claim together with the particular claim $\varphi(0)$ to derive another universal claim, $\forall x\varphi(x)$. 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!

14.5 The induction axiom, the induction rule, the induction schema

The basic idea, we said in §14.3, 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 seems that we

should ideally use a 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. 23. *Induction Axiom:*

$$\forall X([X0 \wedge \forall x(Xx \rightarrow XSx)] \rightarrow \forall xXx)$$

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

Despite that, however, for now we'll concentrate 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 issues about second-order logic. For a start, there are technical issues: it isn't effectively axiomatizable (you can't wrap up the consequence relation for second-order sentences into a nice formalizable logical system). And there are philosophical issues: how well do we understand the idea of quantifying over 'arbitrary properties'? But we can't go into these worries here but will leave that to your core mathematical logic course.

But 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 §14.2. So long as $\varphi(x)$ expresses a kosher property – and we'll say in a moment what that might come to – we can apply the inference rule

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

Alternatively, we can set down the first order

Defn. 24. *Induction Schema:*

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

and then say that for every kosher $\varphi(x)$, the corresponding instance of the induction schema is to be an axiom. Evidently, having this *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 *IGT*, §10.2, and see the idea being put to work in *IGT*, §10.3. We won't fuss about elaborating this point here.

15 First-order Peano Arithmetic

15.1 Getting generous with induction

Suppose then we start again from \mathbb{Q} , and aim to build a richer theory in the language L_A (as defined in §13.1) 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 '0', the familiar successor, addition and multiplication functions, plus identity and other logical apparatus. Therefore – you might very well suppose – it ought also to express a perfectly determinate arithmetical property or relation (even if, in the general case, we can't always decide whether a given number n has the property or not). *So why not be generous and allow any open L_A wff at all to be substituted for φ in the schema?*

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

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

So they actually only allow us to derive some $\forall x\varphi(x)$ when we can *already* prove the corresponding (i) $\varphi(0)$ and also can prove (ii) $\forall x(\varphi(x) \rightarrow \varphi(Sx))$. 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 $\varphi(0)$, $\varphi(S0)$, $\varphi(SS0)$, \dots . 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) $\forall x\varphi(x)$ 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 $\varphi(x)$ has other variables dangling free.)

15.2 Introducing PA

Suppose then that we accept the conclusion of our last argument, and now take it that *any* open wff of L_A can be used in the induction schema. This means moving on from \mathbf{Q} , and jumping right over a range of possible intermediate theories, to adopt the much richer theory of arithmetic that we can briskly define as follows:

Defn. 25. PA – First-order Peano Arithmetic¹³ – is the first-order theory whose language is L_A and whose axioms are those of \mathbf{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

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

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 \mathbf{Q} . So our initial presentation of PA – as explicitly having all the Axioms of \mathbf{Q} plus the instances of the Induction Schema – involves a certain redundancy. Bearing that in mind, here's our \dots

15.3 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:

¹³The 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.

- 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.)

15.4 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 \mathbf{Q} true while making $\forall x(0 + x = x)$ false: that had 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 §13.4) are all provable in PA using induction (and rule out other simple deviant models) There are more than enough examples worked through in *IGT*, 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 \mathbf{P} to be the theory couched in the language L_P , whose axioms are \mathbf{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, \mathbf{P} is PA minus multiplication. *Then \mathbf{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 \mathbf{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 certainly 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 \mathbf{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 episode to explain what that means.

16 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? Sure, there are 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 formal axiomatized 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 will be ludicrously long, far to 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.¹⁴

17 Quantifier complexity

OK, that's the main course over for this episode. Make sure you have fully digested it before moving on to dessert! If you haven't, go back and re-read as necessary. Otherwise, we move on to briefly consider issues about the 'quantifier complexity' of 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 *IGT*, §23.3). 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. Now, we are *not* going to explore these weak arithmetics here. But, irrespective of that, it is in fact worth knowing what Δ_0 , Σ_1 , and Π_1 wffs are. So this section briskly explains.

¹⁴If you know the compactness theorem for first-order theories, you'll probably have met the proof. A reminder: Add to the language of PA a new constant c , and add to the axioms of PA the additional axioms $c \neq 0$, $c \neq \bar{1}$, $c \neq \bar{2}$, ..., $c \neq \bar{n}$, ... Evidently each finite subset of the axioms of the new theory has a model (assuming PA is consistent); just take the standard model of arithmetic and interpret c to be greater than the maximum n for which $c \neq \bar{n}$ is in the given finite suite of axioms. So, by compactness, the infinite set of axioms of the new theory has a model, indeed – by the Löwenheim-Skolem theorem again – a countable model, which contains a zero, its successors, and rogue elements including the denotation of c . This structure is, a fortiori, a non-standard countable model of PA.

17.1 Bounded quantification

As we said before in §13.5, 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 like \mathbf{Q} and \mathbf{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 $\exists v(v + \xi = \zeta)$. And it is standard to further abbreviate such wffs by $(\forall\xi \leq \kappa)\varphi(\xi)$ and $(\exists\xi \leq \kappa)\varphi(\xi)$ respectively.

For any theory T containing \mathbf{Q} – and hence for $T = \mathbf{PA}$ in particular – we have results like these:

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

In other words, theories like \mathbf{Q} and \mathbf{PA} ‘know’ that bounded universal quantifications behave like finite conjunctions, and that bounded existential quantifications behave like finite disjunctions. Hold on to that thought!

17.2 Δ_0 wffs

Let’s informally say that

Defn. 26. *An L_A wff is Δ_0 if its only quantifications are bounded ones.*

For a fancied-up definition, see *IGT*, §9.5. So a Δ_0 wff 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 exactly like a quantifier-free L_A wff, i.e. like an L_B wff, except that we allow ourselves to wrap up some conjunctions like $\varphi(\bar{0}) \wedge \varphi(\bar{1}) \wedge \dots \wedge \varphi(\bar{n})$ into bounded quantifications $(\forall x \leq \bar{n})\varphi(x)$, and similarly wrap up some disjunctions like $\varphi(\bar{0}) \vee \varphi(\bar{1}) \vee \dots \vee \varphi(\bar{n})$ into bounded quantifications $(\exists x \leq \bar{n})\varphi(x)$.

Since 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 equivalent to one, we can mechanically determine the truth-value of a Δ_0 sentence. It follows, of course, that we can mechanically determine whether a Δ_0 open wff $\varphi(x)$ is satisfied by a number n by determining whether $\varphi(\bar{n})$ 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 13 that even \mathbf{Q} can correctly decide all quantifier-free L_A sentences, and \mathbf{Q} knows that *bounded* quantifications behave just like conjunctions/disjunctions, it won’t be a surprise to hear that we have

Theorem 18. *\mathbf{Q} (and hence \mathbf{PA}) can correctly decide all Δ_0 sentences.*

17.3 Σ_1 and Π_1 wffs

We now say, again informally, that

Defn. 27. *An L_A wff is Σ_1 if it is (or is 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 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 ΣxFx – that’s a Greek ‘S’ for ‘(logical) sum’. Likewise the ‘ Π ’ in ‘ Π_1 ’ comes from corresponding symbol for the universal quantifier, as in ΠxFx – 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.¹⁵

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 .

17.4 Two results

Here's another pretty trivial result:

Theorem 19. \mathbf{Q} (and hence PA) can prove any true Σ_1 sentences (is ' Σ_1 -complete').

Proof. Take, for example, a sentence of the type $\exists x\exists y\varphi(x, y)$, where $\varphi(x, y)$ is Δ_0 . If this sentence is true, then for some pair of numbers m, n , the Δ_0 sentence $\varphi(\bar{m}, \bar{n})$ must be true. But then by Theorem 18, \mathbf{Q} proves $\varphi(\bar{m}, \bar{n})$ and hence $\exists x\exists y\varphi(x, y)$, by existential introduction.

Evidently the argument generalizes for any number of initial quantifiers, which shows that \mathbf{Q} proves all truths which are (or are provably-in- \mathbf{Q} equivalent to) some Δ_0 wff preceded by one or more unbounded existential quantifiers. \square

But if that seems straightforward and unexciting, the following corollary is interesting and perhaps initially surprising:

Theorem 20. If T is a consistent theory which includes \mathbf{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 \mathbf{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. \square

This is, in its way, a rather remarkable observation. It means that we don't have to fully *believe* a theory T – i.e. don't have to accept all its theorems are *true* on the interpretation built into T 's language – in order to use it to establish that some Π_1 arithmetic generalization is true. For example, it turns out that, with some trickery, we can state for example Fermat's Last Theorem as a Π_1 sentence. And Andrew Wiles showed how to prove Fermat's Last Theorem using some very heavy-duty infinitary mathematics. Now we see, rather intriguingly, that we don't have to believe that this infinitary mathematics is *true* – whatever exactly that means when things get 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.

Now read *IGT*, chs. 9 and 10.

¹⁵By the same token, a Π_2 wff is one that starts with *two* blocks of quantifiers, a block of universal quantifiers followed by a block of existential quantifiers followed by a bounded kernel. And so it goes.

Primitive recursive functions

Peter Smith

University of Cambridge

November 12, 2010

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

- 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 (as in previous episodes) 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. The next episode shows that L_A , the language of basic arithmetic (see §13.1), can *express* all p.r. functions and relations. Moreover \mathbf{Q} and hence \mathbf{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 \mathbf{PA} , despite having only successor, addition and multiplication ‘built in’, can actually deal with a vast range of functions.

Note, by the way, the link with our earlier talk about “sufficiently strong theories”. Those, recall, are theories that can capture *all* effectively decidable properties of numbers. Well, now we are going to show that \mathbf{PA} (indeed, even \mathbf{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 next.

3. We introduce again the key idea of the ‘arithmetization of syntax’ by Gödel-numbering. Focus on \mathbf{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, 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 $Prf(x, y)$ which captures the relation Prf . And we can 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 . . .

18 Introducing the primitive recursive functions

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

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

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

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

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

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

We’ve seen this two-clause pattern before, of course, in our formal Axioms 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:

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

Three comments about our examples so far:

- i. In each definition, the second clause fixes the value of a function for argument Sn by invoking the value of the *same* function for argument n . This kind of procedure where we evaluate a function by calling the same function is standardly termed ‘recursive’ – or 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.

¹Strictly 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.

- 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:

$$\begin{aligned} P(0) &= 0 \\ P(Sx) &= x \\ x \dot{-} 0 &= x \\ x \dot{-} Sy &= P(x \dot{-} y) \\ |x - y| &= (x \dot{-} y) + (y \dot{-} x) \end{aligned}$$

' 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. 28. *Roughly: a primitive recursive function is one that can be similarly characterized using a chain of definitions by recursion and composition.*²

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.

19 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. 28.

19.1 Definition by primitive recursion – one and two place functions

Consider the recursive definition of the factorial again:

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

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

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

Here, g is just a number, while h is – crucially – a function we are assumed already to know about prior to the definition of f . Maybe 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:

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

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

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

- i. To get the definition of addition to fit this pattern, we have to take $g(x)$ to be the trivial identity function $I(x) = x$.
- ii. To get the definition of multiplication to fit the pattern, $g(x)$ has to be treated as the even more trivial zero function $Z(x) = 0$.
- iii. Again, to get the definition of addition to fit the pattern, we have to take $h(x, y, u)$ to be the function Su . As this illustrates, we must allow h not to care what happens to some of its arguments. One neat way of doing this is to help ourselves to some 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.

19.2 The initial functions

With that motivation, we will now officially define the full ‘starter pack’ of functions as follows:

Defn. 29. *The initial functions are the successor function S , the zero function $Z(x) = 0$ and all the k -place identity functions, $I_i^k(x_1, x_2, \dots, x_k) = x_i$ 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, \dots, x_k onto the i -th axis.

19.3 Definition by primitive recursion – generalizing

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

Defn. 30. *Suppose that the following holds:*

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

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.

19.4 Definition by composition

We need to tidy up the idea of definition by composition. The basic idea, to repeat, is that we form a composite function f by treating the output value(s) of one or more given functions g, g', g'', \dots , 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. 31. *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 $h(g(x, y), g'(y, z))$.

19.5 Putting everything together

We informally defined the primitive recursive (henceforth, p.r.) 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:

Defn. 32. *The p.r. functions are as follows*

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. 33. *A full definition for the p.r. function f is a specification of a sequence of functions $f_0, f_1, f_2, \dots, f_k$ 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 has a full definition in this defined sense. (That’s the sharp version of the informal characterization we gave at the end of §18.)

20 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 way of proving that every p.r. function has some property P . 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 $f_0, f_1, f_2, \dots, f_k$ 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 full definition for f : do you see why? See §14.3.)

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.

21 The p.r. functions are computable

21.1 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:

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

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.

21.2 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. $fact := (fact \times Sy)$
4. Loop

Here $fact$ is a memory register that we initially prime with the value of $0!$. Then the program enters a loop: and the crucial thing about executing a ‘for’ loop is that the total number of iterations to be run through is fixed in advance: we number the loops from 0, and in executing the loop, you increment the counter by one on each cycle. So in this case, on loop number k the program replaces the value in the register with S^k 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

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

with the corresponding program

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

So long as h is already computable, the value of $f(n)$ will be computable using this ‘for’ loop that terminates with the required value in the register $func$.

Similarly, of course, for many-place functions. For example, the value of the two-place function defined by

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

is calculated by the algorithmic program

1. $func := g(m)$
2. For $y = 0$ to $n - 1$
3. $func := h(m, y, func)$
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 21. *Primitive recursive functions are effectively computable by a series of (possibly nested) ‘for’ loops.*

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

Theorem 22. *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.

22 Not all computable numerical functions are p.r.

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

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 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.*

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 23. *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, \dots , such that each of the f_i is p.r., and each p.r. function appears somewhere on the list.

This holds because, by definition, every p.r. function has a ‘recipe’ in which it is defined by recursion or composition from other functions which are defined by recursion or composition from other functions which are defined . . . ultimately in terms of some primitive starter functions. So choose some standard formal specification language for representing these recipes. Then we can

| | | | | | |
|-------|----------------------------|----------------------------|----------------------------|----------------------------|-----|
| | 0 | 1 | 2 | 3 | ... |
| f_0 | <u>$f_0(0)$</u> | $f_0(1)$ | $f_0(2)$ | $f_0(3)$ | ... |
| f_1 | $f_1(0)$ | <u>$f_1(1)$</u> | $f_1(2)$ | $f_1(3)$ | ... |
| f_2 | $f_2(0)$ | $f_2(1)$ | <u>$f_2(2)$</u> | $f_2(3)$ | ... |
| f_3 | $f_3(0)$ | $f_3(1)$ | $f_3(2)$ | <u>$f_3(3)$</u> | ... |
| ... | ... | ... | ... | ... | ↘ |

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.

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

By construction, however, the function δ can’t be primitive recursive. For suppose otherwise. Then δ must appear somewhere in the enumeration of p.r. functions, i.e. be the function f_d for some index number d . But now ask what the value of $\delta(d)$ is. By hypothesis, the function δ is none other than the function f_d , so $\delta(d) = f_d(d)$. But by the initial definition of the diagonal function, $\delta(d) = f_d(d) + 1$. Contradiction.

So we have ‘diagonalized out’ of the class of p.r. functions to get a new function δ which is effectively computable but not primitive recursive. □

‘But hold on! *Why* is the diagonal function not a p.r. function? Where are the open-ended searches involved in computing it?’ Well, consider evaluating $\delta(n)$ for increasing values of n .

For each new argument, we will have to look along the sequence of strings of symbols in our computing language until we find the next one that gives us a well-constructed recipe for a p.r. function. That isn't given to us a bounded search.

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

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

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

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. 35. *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.*

Given that any p.r. function is effectively computable, p.r. decidable properties and relations are among the effectively decidable ones. Hence the appropriateness of the label!

Now read *IGT*, Ch. 11.

Expressing and capturing the primitive recursive functions

Peter Smith

University of Cambridge

November 19, 2010

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

- 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.
-

A. 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!

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. 30).

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. 32).

We then noted three key facts:

1. Every p.r. function is intuitively computable – moreover it is computable without going in for open-ended searches. It can be computed using only 'for' loops and not open-ended 'do until' loops. That's Theorem 21.
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 22.
3. But not every intuitively computable numerical function is primitive recursive. That's Theorem 23.

B. Let's just comment, by way of an aside, on the proof of the third result.

We noted that we can effectively list (the recipes for) p.r. functions, specifying the functions f_0, f_1, f_2, \dots . We can then define the function $d(n) = f_n(n) + 1$. This is effectively computable, because we just go along the effectively generated list of recipes till the n -th one, and use that recipe applied to input n to compute $f_n(n)$ and then add one. But this computable function is distinct from all the f_j . So it isn't primitive recursive.

The argument evidently generalizes. Suppose we can effectively list the recipes for some other class C of computable total (i.e. everywhere-defined) functions, specifying the functions $f_0^C, f_1^C, f_2^C, \dots$. Then again we can define $d^C(n) = f_n^C(n) + 1$, which will be everywhere-defined because f_j^C is everywhere defined, is computable, but not in C . In a slogan, we can ‘diagonalize out’ of class C . So that gives us a theorem:

Theorem 24. *No effective listing of algorithms can include algorithms for all the intuitively computable total functions.*¹

C. OK, so the situation is now this. We’ve been talking about *formal* arithmetics with just three functions – successor, addition, multiplication – built in. We’ve reminded ourselves that ordinary *informal* arithmetic talks about heaps more elementary functions like the exponential, the factorial, the-number-of-prime-divisors-of, 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.* 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. 16. 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 claiming is that we can show at least that Q and hence PA can capture all primitive recursive computable functions. That will be enough for Gödel’s argument for incompleteness to fly.

D. One more remark before we set off. The main proof(-sketch) in this episode is not particularly difficult, but it is a bit messy. Don’t worry at all, however, if you find it troublesome to see what is going on. *Nothing in what follows in later episodes depends on your knowing the details of the proofs of Theorems 25 and 27.* All you need to know is that the results can be proved. So don’t let this episode be a stumbling block!

24 L_A can express all p.r. functions

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

24.1 Proof strategy

Suppose that the following three propositions are all true:

- E1. L_A can express the initial functions, and addition and multiplication. (See Defn. 29.)
- 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. 31.)
- 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. 30.)

Then by the argument of §20, that establishes

Theorem 25. *L_A can express all p.r. functions.*

¹Note 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. $\delta(n) = \varphi_n(n) + 1$ could then be e.g. φ_d , if $\varphi_d(d)$ and hence $\varphi_d(d) + 1$ are both undefined.

But it is trivial to prove E1. Just look at cases. The successor function $Sx = y$ is of course expressed by the open wff $Sx = y$. The addition function $x + y = z$ is expressed by $x + y = z$. Similarly for multiplication.

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

Finally, the three-place identity function $I_2^3(x, y, z) = y$, to take just one example, is expressed by the wff $I_2^3(x, y, z, u) =_{\text{def}} (x = x \wedge y = u \wedge z = z)$. Likewise for all the other identity functions. [Check those claims!]

So that just leaves E2 and E3 to prove.

24.2 Proving E2

This result is pretty trivial too. Suppose g and h are one-place functions, expressed by the wffs $G(x, y)$ and $H(x, y)$ respectively. Then, the function $f(x) = h(g(x))$ is evidently expressed by the wff $\exists z(G(x, z) \wedge H(z, y))$.

For suppose $g(m) = k$ and $h(k) = n$, so $f(m) = n$. Then by hypothesis $G(\bar{m}, \bar{k})$ and $H(\bar{k}, \bar{n})$ will be true, and hence $\exists z(G(\bar{m}, z) \wedge H(z, \bar{n}))$ is true, as required. Conversely, suppose $\exists z(G(\bar{m}, z) \wedge H(z, \bar{n}))$ is true. Then since the quantifiers run over numbers, $(G(\bar{m}, \bar{k}) \wedge H(\bar{k}, \bar{n}))$ 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.

24.3 What it takes to define the factorial

Proving E3 is the tricky case.² We'll illustrate the general strategy by first taking a particular case of a definition by primitive recursion, and then we'll generalize. So consider the primitive recursive definition of the factorial function again:

$$\begin{aligned} 0! &= 1 \\ (Sx)! &= x! \times Sx \end{aligned}$$

The multiplication and successor functions here are of course expressible in L_A : but how can we express our defined function in L_A ?

Think about the p.r. definition for the factorial in the following way. It tells us how to construct a sequence of numbers $0!, 1!, 2!, \dots, x!$, 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 thus says

- A. There is a sequence of numbers k_0, k_1, \dots, 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 a decoding function *decode* such that if you feed *decode* the code number c and the index i it 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, \dots, k_x we want: $\text{decode}(c, i) = k_i$.

If we can find a coding scheme and a decoding function that does the trick, then we can rewrite the p.r. definition of the factorial as

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

And if *decode* can be expressed in L_A then we can define the factorial in L_A .

²Don't worry if you find the ensuing argument a bit boggling (though really you shouldn't, as the basic proof idea is not hard even if its implementation takes a bit of a trick). As far as understanding Gödel's theorems are concerned, what you really need to know is that Theorem 25 *can* be proved, and not the details about *how* it is proved.

24.4 Coding sequences if we have a factorizing function to play with

Let's just note – before giving a decode function that can be expressed in pure L_A – that if we were working in a slightly richer language the task would be easy.

Suppose $\pi_0, \pi_1, \pi_2, \pi_3, \dots$ is the series of prime numbers 2, 3, 5, 7, \dots . Now consider the number

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

b can be thought of as encoding the whole sequence $k_0, k_1, k_2, \dots, k_n$. And we can recover the coded sequence from b by using the (primitive recursive) decoding function *exponent* where $\text{exponent}(b, i)$ is the exponent of the i -th prime in the prime factorization of b . (That's unique by the fundamental theorem of arithmetic that says that prime factorizations are unique.)

So *there is nothing at all mysterious about a coding scheme for finite sequences and a decoding function that recovers elements of the sequence from its code*: the decoding function *exponent* would do the job. And a language L_A^+ with an expression for *exponent* built in would be able to define the factorial function in the way explained.

But of course, *exponent* isn't built into L_A or obviously definable in it. The question now is: can we construct a different coding scheme with a decoding function which can be constructed from the successor, addition and multiplication functions which *are* built into L_A ?

24.5 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 $\text{decode}(c, d, i)$ such that, for *any* finite sequence of natural numbers $k_0, k_1, k_2, \dots, k_n$ there is a pair of code numbers c, d such that for every $i \leq n$, $\text{decode}(c, d, i) = k_i$.

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

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

Then, given any sequence k_0, k_1, \dots, 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 (mathmos: see *IGT*, 13.4, fn. 6 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:

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

This, as we want, expresses our Gödelian β -function in L_A (for remember, we can define ' \leq ' in L_A).

24.6 Defining the factorial in L_A

We've just claimed: given any sequence of numbers k_0, k_1, \dots, k_x , there are code numbers c, d such that for $i \leq x$, $\beta(c, d, i) = k_i$. So we can reformulate

- A. There is a sequence of numbers k_0, k_1, \dots, k_x such that: $k_0 = 1$, and if $u < x$ then $k_{Su} = k_u \times Su$, and $k_x = y$,

as follows:

B'. There is some pair 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 translate (C) into L_A as follows:

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

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

24.7 Generalizing to prove E3

Finally, we need to show that we can use the same β -function trick and prove more generally that, if the function f is defined by recursion from functions g and h which are already expressible in L_A , then f is also expressible in L_A .

So here, just for the record, is the entirely routine generalization we need (there are no new ideas here – just unavoidable clutter).

We are assuming that

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

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, \dots, 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{x} ' to indicate a suitable sequence of n variables – (B*) can be rendered into L_A by

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

Abbreviate this defined wff $\varphi(\vec{x}, y, z)$; 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 from the start of §24.1. Hence every p.r. function can be expressed in L_A .

Theorem 25 is in the bag!

25 Primitive recursive functions can be canonically expressed by Σ_1 wffs

In this section, we extract more information out of the proof of Theorem 25. In particular we show that the L_A wff needed to express a p.r. function is logically not very complex – a Σ_1 wff (in the sense of Defn. 27) is enough to do the job.

25.1 The proof of Theorem 25 is constructive

What we 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 .

Remember: a full definition for f is in effect a recipe for defining a whole sequence of functions $f_0, f_1, f_2, \dots, f_k$ 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. In the terms of §24.1, 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.

For brevity, let’s now say that

Defn. 36. *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 25.*

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

The canonical wff which reflects a full definition of f is built up starting from wffs expressing initial wffs (and addition and multiplication). 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 $G(x, y)$ and $H(x, y)$ respectively. Then, the function $f(x) = h(g(x))$ is expressed by the wff $\exists z(G(x, z) \wedge H(z, y))$ 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 two-place function h by the *three*-variable Σ_1 expression H . Then if f is defined from g and h by primitive recursion, f can be expressed by

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

And this too is Σ_1 . For B is Σ_1 : 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* 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 26. *L_A can express any p.r. function f by a Σ_1 canonical wff which recapitulates a full definition for f .*

26 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 also:

Theorem 27. *The theory Q can capture any p.r. function by a Σ_1 wff.*

Recall, ‘capturing’ a function here means being able to case-by-case prove formulae that in effect assign the right values to the function (see Defn. 15). So the formula $\chi(x, y)$ captures the one-place function f in Q if, when $f(m) = n$, $Q \vdash \chi(\bar{m}, \bar{n})$, and when $f(m) \neq n$, $Q \vdash \neg\chi(\bar{m}, \bar{n})$. Similarly for many-place functions.

Now there's more than one route to Theorem 27. I'll mention *the direct assault* and *the clever trick*. In *IGT*, Ch. 13, I go 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. And this is just to once more replicate the overall strategy for proving results about all p.r. functions which we described in §20, and deployed already in §21.1 and §24.1.

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 §24.1 – 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 §24.1 *express* the initial functions in fact serve to *capture* the initial functions in Q.

How do we prove C2? Again we track the proof in §24.2. Suppose g and h are one-place functions, captured by the wffs $G(x, y)$ and $H(x, y)$ respectively. Then we prove that the function $f(x) = h(g(x))$ is captured by the wff $\exists z(G(x, z) \wedge H(z, y))$ (which is Σ_1 if G and H are).

And how do we prove C3? This is the tedious case that takes hard work! We need to show the formula B not only expresses but captures Gödel's β -function. 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 D^* in §24.7 captures the function f defined by primitive recursion from g and h . (If you want the gory details establishing that, then you can consult e.g. Elliott Mendelson, *Introduction to Mathematical Logic*, 4th edn, Prop. 3.24. And again, the result is Σ_1 if G and H are.)

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 §24 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 .

27 Expressing/capturing properties and relations

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

We said in §23, Defn. 34 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 25, L_A can express c_P by a two-place open wff $c_P(x, y)$. So if m is P , then $c_P(m) = 0$, then $c_P(\bar{m}, 0)$ is true. And if m is not P , then $c_P(m) \neq 0$, then $c_P(\bar{m}, 0)$ is not true. So, by the definition of expressing-a-property, the wff $c_P(x, 0)$ 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 25 that

Theorem 28. *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 27, Q can capture c_P by a two-place open wff $c_P(x, y)$. So if m is P , then $c_P(m) = 0$, so $Q \vdash c_P(\bar{m}, 0)$. And if m is not P , then $c_P(m) \neq 0$, then $Q \vdash \neg c_P(\bar{m}, 0)$. So, by the definition of capturing-a-property, the wff $c_P(x, 0)$ 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 27 that

Theorem 29. *Q can capture all p.r. decidable properties and relations.*

Now read *IGT*, Chap 13.

Gödel Without (Too Many) Tears – 7

The arithmetization of syntax

Peter Smith

University of Cambridge

January 14, 2011

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

- 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 §28 is more scene-setting which (depending on your background) you might well just want to skip through. The real action continues with §29 on p. 5.

Anyway, let’s take stock yet again.

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, 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 what 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 $\forall x(0 + x = x)$. [Reality check: how is that BA is complete and yet the stronger theory Q isn’t?]
4. Q, as we said, is boringly incomplete, even for just the arithmetic of successor, addition and multiplication. How can we beef it up at least so that it can prove elementary quantified truths like $\forall x(0 + x = x)$? 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. 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 5, we looked at a whole family of functions – the primitive recursive functions – which can be 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.

Episodes 3 to 5 can be thought of as very important scene-setting: but our real hard work proving (versions of) the First Incompleteness Theorem starts with ...

6. Episode 6 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 \mathbf{Q} (and hence, a fortiori \mathbf{PA}) can *capture* all the p.r. functions: if we construct the wff φ that expresses f in the right way, then if $f(m) = n$, $\mathbf{Q} \vdash \varphi(\bar{m}, \bar{n})$, and if $f(m) \neq n$, $\mathbf{Q} \vdash \neg\varphi(\bar{m}, \bar{n})$.

Now that last result in Episode 6 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 6 shows that \mathbf{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 \mathbf{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. \mathbf{Q} 's being able to capture all p.r. functions is enough to make it the case that sensibly axiomatized theories that contain \mathbf{Q} are incomplete (so long as they are consistent and satisfy another modest condition).

Now read on ...

28 Formalization and finitary reasoning

28.1 Formalization and axiomatization again

Before proceeding, let's give ourselves a few gentle reminders about the business of *formalization* – something that is now familiar to any logic student.

In elementary logic classes, we are drilled in translating arguments into an appropriate formal language and then constructing formal deductions of putative conclusions from given premisses. Why bother with formal languages? Because everyday language – even mathematical English – is replete with redundancies and ambiguities. So, in assessing complex arguments, it helps to regiment them into a suitable artificial language which is expressly designed to be free from obscurities, and where surface form reveals logical structure. And why bother with formal deductions? Because informal arguments – even mathematical arguments – often involve suppressed premisses (and there is the lurking danger of inferential fallacies). It is only too easy to cheat. Setting out arguments as formal deductions in one style or another enforces honesty: we have to keep a tally of the premisses we invoke, and of exactly what inferential moves we are using. And honesty is the best policy. For suppose things go well with a particular formal deduction. Suppose we get from the given premisses to some target conclusion by small inference steps each one of which is obviously valid (no suppressed premisses are smuggled in, and there are no suspect inferential moves). Our honest toil then buys us the right to confidence that our premisses really do entail the desired conclusion.

Granted, outside the logic classroom we almost never set out deductive arguments in fully formalized versions. No matter. We have glimpsed a first ideal – arguments presented in an entirely perspicuous language with maximal clarity and with everything entirely open and above board,

leaving no room for misunderstanding, and with all the arguments' commitments systematically and frankly acknowledged.

Old-fashioned presentations of Euclidean geometry illustrate the pursuit of a related second ideal – the (informal) axiomatized theory. Like beginning logic students, school students used to be drilled in providing deductions, though the deductions were framed in ordinary geometric language. The game is to establish a whole body of theorems about (say) triangles inscribed in circles, by deriving them from simpler results, which had earlier been derived from still simpler theorems that could ultimately be established by appeal to some small stock of fundamental principles or axioms. And the aim of this enterprise? By setting out the derivations of our various theorems in a laborious step-by-step style – where each small move is warranted by simple inferences from propositions that have already been proved – we develop a unified body of results that we can be confident must hold if the initial Euclidean axioms are true. On the surface, school geometry perhaps doesn't seem very deep: yet making all its fundamental assumptions fully explicit is surprisingly difficult. And giving a set of axioms invites further enquiry into what might happen if we tinker with these assumptions in various ways – leading, as is now familiar, to investigations of non-Euclidean geometries.

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

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

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

28.2 Responding to paradox

Think yourself back to the situation in mathematics 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 storable 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. They have loosed themselves 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 pile up. What to do?

We might distinguish two kinds of responses that we can have to the paradoxes that threaten Cantor’s paradise where mathematicians can play freely, what we might suggestively call the *foundationalist* and the ‘*more mathematics*’ lines.

Foundationalist responses to paradox Consider first the option of seeking “foundations”. We could, for example, 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 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 structure of the natural world is being shaken.

So put the option of anchoring mathematics in the physical world aside. But perhaps (i) we could try to ensure that our mathematical constructions are grounded in mental constructions that we can perform and have a secure epistemic access to. Or alternatively (ii) we could try to go back to find incontrovertible logical principles and definitions of mathematical notions in logical terms, and try to constrain mathematics to what we can reconstruct on a firm logical footing.

Response (i) leads us to intuitionism. But it depends on an obscure notion of mental construction, and in any case – in its most worked out form – cripples lots of classical mathematics that we thought was unproblematically in good order, rather than giving it a foundation.

What about the logicist response (ii)? This 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 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: let’s see if we can write a *Principia Mathematica* . . . !

‘Mathematical’ responses to paradox Maybe, however, we just shouldn’t be seeking to give mathematics a philosophical “foundation”. After all, the paradoxes arise within mathematics, and to avoid them (the working mathematician might reasonably think) we just need to do mathematics more carefully. As Peirce – for example – held, mathematics risks being radically distorted if we seek to make it answerable to some outside considerations (whether from philosophy or logic). And we don’t need to look outside 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 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 perhaps this axiomatized theory *is* consistent and trouble-free?

Well, to explore this question, the thought goes, requires not ‘looking outside’ mathematics for foundations, but engaging in more mathematical enquiry.

Note that both the foundationalist logicist and the ‘do maths better’ camp, are keen on battering a mathematical theory T into nice tidy axiomatized formats and sharply defining the rules of the game. But for a foundationalist logicist this is a step on the way to showing T is *true-because-safely-derivable-from-truisms*. The ‘do maths better’ camp just want axiomatization to be a prelude to giving us enough control over theory T that we can show it is *safe-to-work-with-because-consistent*.

28.3 Applying finitary mathematics to formal theories

Now note a key observation emphasized by Hilbert.

The axiomatic formalization of a mathematical theory T about widgets, wombats, or whatever, 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 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, you can see the attractions of Hilbert’s hopeful programme here – the programme of showing various systems of infinitary mathematics are contradiction-free by giving finitary consistency proofs.

And now, enter Gödel . . .

29 Arithmetization

We just noted Hilbert’s insight 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 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.

29.1 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 will work for any formal language). There are of course different ways of doing this: we’ll follow in style Gödel and tradition!

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 | \dots |
| 1 | 3 | 5 | 7 | 9 | 11 | 13 | 15 | 17 | 19 | 21 | 23 | 25 | 27 | 2 | 4 | 6 | \dots |

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

Defn. 37. Let expression e be the sequence of $k + 1$ symbols and/or variables $s_0, s_1, s_2, \dots, s_k$. 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 $2^{c_0} \cdot 3^{c_1} \cdot 5^{c_2} \cdot \dots \cdot \pi_k^{c_k}$.

For example:

- i. The single symbol 'S' has the g.n. 2^{23} (the first prime raised to the appropriate power as read off from our correlation table of basic codes).
- ii. The standard numeral SS0 has the g.n. $2^{23} \cdot 3^{23} \cdot 5^{21}$ (the product of the first three primes raised to the appropriate powers).
- iii. The wff

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

has the g.n.

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

That last number is, of course, *enormous*. So when we say that it is elementary to decode the resulting g.n. by taking the exponents of prime factors, we don't mean that the computation is quick. 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.

29.2 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. And in this rather old-fashioned framework, proof-arrays are simply *linear sequences* of wffs. A nice way of coding these is by what we'll call *super Gödel numbers*.

Defn. 38. Given a sequence of wffs or other expressions $e_0, e_1, e_2, \dots, e_n$, we first code each e_i by a regular g.n. g_i , to yield a sequence of numbers $g_0, g_1, g_2, \dots, g_n$. 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 $2^{g_0} \cdot 3^{g_1} \cdot 5^{g_2} \cdot \dots \cdot \pi_n^{g_n}$.

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.

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

For this subsection, we'll continue to focus on the language L_A . But similar remarks will apply mutatis mutandis to any sensibly built formal language. We begin with some definitions (see §4):

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 30. *$Term(n)$, $Wff(n)$, and $Sent(n)$ are p.r. decidable properties.*

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

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

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. 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 without unbounded searches, 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, without any open-ended searches. And again, what's computably decidable without open-ended searches is primitive-recursively decidable. \square

29.4 *Prf* is a p.r. relation

In this subsection, we'll focus on the theory PA. But again, similar remarks will apply mutatis mutandis to any sensibly axiomatized theory.

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 31. *$Prf(m, n)$ 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: 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 doesn't involve any open-ended search. \square

A similar result, as we said will hold for the corresponding Prf_T relation for any theory T which is like PA in this respect: we can mechanically check whether a string of wffs constitutes a T -proof without having to engage in any open-ended search. 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.)

29.5 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 algorithms don't involve open-ended computations using unbounded searches: in other words, the computations can be done just using 'for' loops.

So let S be any other comparable coding scheme, which similarly involves a pair of algorithmic methods for moving to and fro between L_A expressions and numerical codes (where the methods don't involve open-ended searches). And suppose S assigns code n_1 to a certain L_A expression. Consider the process of first decoding n_1 to find the original L_A expression and then re-encoding the expression using our Gödelian scheme to get the code number n_2 (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 32. 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). \square

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*.

30 Some cute notation

There's one other main idea we need in the next episode and which we'll introduce in this, namely the idea of 'diagonalization'. But before explaining that in §32, let's pause to introduce a really pretty bit of notation.

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

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 ' $\ulcorner U \urcorner$ ' 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. ' $SS0$ ' is an L_A expression, the standard numeral for 2.
2. On our numbering scheme ' $\ulcorner SS0 \urcorner$ ', the g.n. of ' $SS0$ ', is $2^{21} \cdot 3^{21} \cdot 5^{19}$.
3. So, by our further convention in the book, we can also use the expression ' $\ulcorner SS0 \urcorner$ ' inside (a definitional extension of) L_A , as an abbreviation for the standard numeral for that g.n., i.e. as an abbreviation for ' $SSS \dots S0$ ' 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 ' $\ulcorner SS0 \urcorner$ ' by the slightly messy ' $\overline{\ulcorner SS0 \urcorner}$ '. 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 .

31 Consistency sentences – a very first look

Consider the Prf_Z relation for e.g. ZFC set theory. This, like the corresponding relation for any formalized theory is p.r.: so e.g. PA can capture this relation using a two-place predicate Prf_Z .

Now consider the wff $\forall x \neg Prf_Z(x, \ulcorner \perp \urcorner)$. This arithmetical sentence is true if no number numbers a proof-in-ZFC of absurdity. So the arithmetical sentence 'says' that ZFC is consistent.

If PA can prove it, then we would have an arithmetic proof of the consistency of standard set theory. Even better, if we can do this in a theory weaker than PA: the weaker the resources needed to run the arithmetic consistency proof, the better for Hilbert's programme. So *can* PA prove it?

Can you stand the suspense?

The First Incompleteness Theorem

Peter Smith

University of Cambridge

January 15, 2011

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

- 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 the background that needs 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. (§29)
- 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 ‘ $\ulcorner\varphi\urcorner$ ’ 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. (§30)
- 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. (§29.4)
- 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). (§24)
- 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). (§26)

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

32 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 correlated with wffs.

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

This substitution operation is called *diagonalization*, which at first sight might seem an odd term for it. But in fact, Gödel’s construction involves something quite closely akin to the ‘diagonal’ construction we encountered in §7, where we matched the index of a wff $\varphi_n(x)$ (in an enumeration of wffs with one free variable) with the numeral substituted for its free variable, to form $\varphi_n(\bar{n})$. Here, in our Gödelian diagonal construction, we match U ’s Gödel number – and we can think of this as indexing the wff in a list of wffs – with the numeral substituted for its free variable, and this will yield the Gödel sentence $U(\ulcorner U \urcorner)$.

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

Defn. 44. *The diagonalization of φ is $\exists y(y = \ulcorner \varphi \urcorner \wedge \varphi)$.*

It should go without saying that there is no special significance to using the variable ‘ y ’ for the relevant variable here! But we’ll keep this choice fixed, simply for convenience.

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

Theorem 33. *There is a p.r. function $diag(n)$ which, when applied to a number n which is the g.n. of some wff, yields the g.n. of that wff’s diagonalization.*

Proof. Consider this procedure. Decode the g.n. $n = \ulcorner \varphi \urcorner$ to get some expression φ (assume we have some convention for dealing with ‘waste’ cases where we don’t get an expression). Then form φ ’s diagonalization, $\exists y(y = \ulcorner \varphi \urcorner \wedge \varphi)$. 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 \square

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

Proof. Either we can informally note that we can mechanically check whether $Gdl(m, n)$ holds without open-ended searches.

Or we can note that $Gdl(m, n)$ holds, by definition, when $Prf(m, diag(n))$. The characteristic function of Gdl is therefore definable by composition from the characteristic function of Prf and the function $diag$, and hence is p.r., given facts (iv) and (v) from the preamble. \square

Now Gdl can be expressed in L_A by a Σ_1 wff (by fact vi), which in fact captures Gdl in PA (by fact vii). Of course there won’t be a unique such Σ_1 wff. For a start, there will be more than one way of constructing a full definition of the (characteristic function) for the p.r. relation Gdl , so more than one way of tracking such a definition. But we’ll adopt the following definition:

Defn. 46. *$Gdl(x, y)$ 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. $U(y) =_{\text{def}} \forall x \neg \text{Gdl}(x, y)$.

(For reasons that will become clear in just a moment, you can think of that U as standing for ‘unprovable’.) And now we diagonalize U , to give

Defn. 48. $G =_{\text{def}} \exists y (y = \ulcorner U \urcorner \wedge U(y))$.

Trivially, G is equivalent to $U(\ulcorner U \urcorner)$. Or unpacking that a bit, G is equivalent to $\forall x \neg \text{Gdl}(x, \ulcorner U \urcorner)$.

G – meaning of course the L_A sentence you get when you unpack the abbreviations! – is 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. In the terminology of §17, we have the easy result that

Theorem 35. G is Π_1 .

Proof. $\text{Gdl}(x, y)$ is Σ_1 . So $\text{Gdl}(x, \ulcorner U \urcorner)$ is Σ_1 . So its negation $\neg \text{Gdl}(x, \ulcorner U \urcorner)$ is Π_1 . Hence $\forall x \neg \text{Gdl}(x, \ulcorner U \urcorner)$ is Π_1 too. Its logical equivalent G is therefore also Π_1 . \square

And now the key observation:

Theorem 36. 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 $Gdl(m, \ulcorner U \urcorner)$. 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. $\ulcorner U \urcorner$. But the wff with g.n. $\ulcorner U \urcorner$ 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. \square

34 The First Theorem – the semantic version

34.1 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 $\neg G$ must be false. Hence $\neg G$ cannot be proved in PA either, supposing PA is sound. In Gödel’s words, G is a ‘formally undecidable’ sentence of PA (see Defn. 6).

Which establishes

Theorem 37. If PA is sound, then there is a true Π_1 sentence G such that $PA \not\vdash G$ and $PA \not\vdash \neg G$, 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 .

34.2 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 without going off on an open-ended search. 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. 8), 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 38. *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 $T \not\vdash G_T$ and $T \not\vdash \neg G_T$, so T is negation incomplete.*

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

34.3 Comparisons

Compare Theorem 38 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 $T \not\vdash G_T$ and $T \not\vdash \neg G_T$, 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 Goldbach's conjecture that every number is such that, if even and greater than two, it is the sum of two primes (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 formalized theories more generally. But that's not much loss. For what would a theory look like that was 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 axiomatized theory will be p.r. axiomatized. (We can in fact extend Gödel's theorem to cover the case of abnormally though still decidable axiomatized theories – but that really is a minor extension.)

34.4 Our Incompleteness Theorem is better called an *incompleteness* theorem

Here, we just repeat the argument of §3.2: 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 38 we can find a true G_T such that $T \not\vdash G_T$ and $T \not\vdash \neg G_T$. 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 $T' = T + G_T$. 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 still a p.r. axiomatized theory, since adding an specified axiom to T doesn’t commit us to any open-ended searches to determine what is an axiom of the augmented theory. (iii) We haven’t changed the language. So our Incompleteness Theorem applies, and we can find a sentence $G_{T'}$ such that $T' \not\vdash G_{T'}$ and $T' \not\vdash \neg G_{T'}$. And since T' is stronger than T , we have a fortiori, $T \not\vdash G_{T'}$ and $T \not\vdash \neg G_{T'}$. 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 true 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*.

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 iff and only if unprovable? Do you understand why 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 . . .

35 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 $\forall x\varphi(x)$ by $\forall x(\mathbf{N}x \rightarrow \varphi(x))$, where ‘ \mathbf{N} ’ picks out the numbers from the domain of the theory’s native quantifiers (see Defn. 8).

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

We saw in §13.3 that \mathbf{Q} is ω -incomplete: that’s because it can prove each instance of $0 + \bar{n} = \bar{n}$, but can’t prove $\forall x(0 + x = x)$. We could repair ω -incompleteness if we could add the ω -rule (see §14.1), 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 \mathbf{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 $\varphi(x)$, T can prove each $\varphi(\bar{n})$ and T can also prove $\neg\forall x\varphi(x)$.*

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

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 $\varphi(\bar{n})$ for each m . T is ω -incomplete if it can’t also prove something we’d like it to prove, namely $\forall x\varphi(x)$. 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 $\neg\forall x\varphi(x)$.

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 $\varphi(\bar{n})$ and yet also prove $\neg\forall x\varphi(x)$ 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 $\varphi(x)$, T does prove each of $\varphi(0)$, $\varphi(\bar{1})$, $\varphi(\bar{2})$, \dots . By hypothesis, these theorems will then be true on the given standard interpretation; so this means that every natural number must satisfy $\varphi(x)$; so $\forall x\varphi(x)$ is true since the domain contains only natural numbers. Hence $\neg\forall x\varphi(x)$ will have to be false on this standard interpretation. Therefore $\neg\forall x\varphi(x)$ 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.

36 The First Theorem – the syntactic version

36.1 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, $\text{PA} \not\vdash \text{G}$.*

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, $\text{Gdl}(m, \ulcorner \text{U} \urcorner)$.

Now we use the fact that Gdl captures the relation *Gdl*. That implies that, since $\text{Gdl}(m, \ulcorner \text{U} \urcorner)$, we have (i) $\text{PA} \vdash \text{Gdl}(\bar{m}, \ulcorner \text{U} \urcorner)$.

But since G is logically equivalent to $\forall x \neg \text{Gdl}(x, \overline{\ulcorner \text{U} \urcorner})$, the assumption that G is provable comes to this: $\text{PA} \vdash \forall x \neg \text{Gdl}(x, \ulcorner \text{U} \urcorner)$. The universal quantification here entails any instance. Hence (ii) $\text{PA} \vdash \neg \text{Gdl}(\bar{m}, \ulcorner \text{U} \urcorner)$.

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. \square

36.2 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 $\text{PA} \not\vdash \text{G}$, i.e.,

1. $\text{PA} \not\vdash \forall x \neg \text{Gdl}(x, \overline{\ulcorner \text{U} \urcorner})$.

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 $\text{Gdl}(m, \ulcorner \text{U} \urcorner)$. Hence, again by the fact that Gdl captures *Gdl*, we have

2. For each m , $\text{PA} \vdash \neg \text{Gdl}(\bar{m}, \overline{\ulcorner \text{U} \urcorner})$.

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

36.3 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, $PA \not\vdash \neg G$.*

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

1. $PA \vdash \exists x Gdl(x, \ulcorner U \urcorner)$.

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 $Gdl(m, \ulcorner U \urcorner)$ is false, for each m . So, by the requirement that Gdl captures Gdl , we have

2. $PA \vdash \neg Gdl(\bar{m}, \ulcorner U \urcorner)$ 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. \square

36.4 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). And we know from Theorem 36 that G is true if and only if it is unprovable. 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 $PA \not\vdash G$, and if PA is ω -consistent $PA \not\vdash \neg G$, so – assuming ω -consistency and hence consistency – PA is negation incomplete.*

36.5 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 Gdl_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 $T \not\vdash G_T$, and if T is ω -consistent, $T \not\vdash \neg G_T$, 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 $\varphi, \neg\varphi$ 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 having semantic content.)

36.6 Comparisons

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 property that any sensible formalized arithmetic will share), there is a sentence of basic arithmetic G_T such that $T \not\vdash G_T$ and $T \not\vdash \neg G_T$, 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 that that’s not much loss. (And we can in fact go on to make up the shortfall).

37 The historical First Theorem

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 $T \not\vdash \varphi$, and if T is ω -consistent then $T \not\vdash \neg\varphi$.*

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*.¹

For 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 §34.3). 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 .

¹Hold on! If *that’s* the First Theorem, we didn’t need to do all the hard work showing that Q and PA are p.r. adequate, did we? Well, yes and no. No, proving *this* original version of the Theorem of course doesn’t depend on proving that any particular theory is p.r. adequate. But yes, showing that this Theorem has real bite, showing that it applies to familiar arithmetics, does depend on proving the adequacy theorem.

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. $\forall xF(x)$], 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.

At this point, make sure you really understand at least what the core theorems in this episode *mean*. Then read *IGT*, Chs. 16 and 17.

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

Then when you feel reasonably confident of the techie details, have a look at Ch. 18 (perhaps skipping §18.3).

The Diagonalization Lemma, Rosser and Tarski

Peter Smith

University of Cambridge

February 10, 2011

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

- Provability predicates
 - The Diagonalization Lemma
 - 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. If T is the right kind of ω -consistent theory including enough arithmetic, then there will be an arithmetic sentence G_T such that $T \not\vdash G_T$ and $T \not\vdash \neg G_T$. Moreover, G_T is constructed so that it is true if and only if unprovable-in T (so it is true). We won't rehearse the construction and the arguments from the last episode again here.

This episode starts by proving the key 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.

38 Provability predicates

Recall that, for a p.r. axiomatized theory T , $Prf_T(m, n)$ 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 (see §29.4). Assuming T extends Q , T can capture any p.r. decidable relation, including Prf_T (§26). So we can legitimately stipulate

Defn. 52. $Prf_T(x, y)$ stands in for a T -wff that canonically captures Prf_T ,

for there will indeed be such a wff. NB, for some of what follows, *any* wff Prf_T that captures Prf_T will do (it doesn't have to be 'canonical' in the sense of Defn. 36): but it's convenient to fix on some canonical way, hence Σ_1 way, of capturing Prf_T . Next, we say:

Defn. 53. Put $Prov_T(y) =_{\text{def}} \exists v Prf_T(v, y)$: such an expression is a provability predicate for T .

$Prov_T(\bar{n})$ 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 $Prov_T(\ulcorner \varphi \urcorner)$ is true just when φ is a theorem. Hence the aptness of the label 'provability predicate' for $Prov_T$. Note, if $Prov_T$ is built from a canonical Prf_T it is also Σ_1 .

So our observation that G_T is so constructed that it is true if and only if unprovable-in T can in fact be *expressed* inside T itself, by the wff $G_T \leftrightarrow \neg Prov_T(\ulcorner G_T \urcorner)$.

And now a key observation: T doesn't just express this fact but can quite easily *prove* it too, i.e. we have $T \vdash G_T \leftrightarrow \neg Prov_T(\ulcorner G_T \urcorner)$. More on this claim below, §40.3.

39 The Diagonalization Lemma

39.1 Introducing the Lemma

Now, when we think through a demonstration that there is some sentence δ (in fact our old friend G_T) such that $T \vdash \delta \leftrightarrow \neg \text{Prov}_T(\ulcorner \delta \urcorner)$, we notice an interesting generalization. *Take any open sentence $\varphi(x)$ at all – i.e. not just $\neg \text{Prov}_T(x)$ – then we can always find some sentence δ or other such that $T \vdash \delta \leftrightarrow \varphi(\ulcorner \delta \urcorner)$.* This generalized result is called The Diagonalization Lemma (and was first explicitly isolated by Carnap as a principle that could be seen as underlying Gödel’s incompleteness proof).

39.2 Proving the Lemma

First a reminder of something familiar:

Defn. 15. *The theory T captures the one-place function f by the open wff $\varphi(x, y)$ iff, for any m, n ,*

- i. if $f(m) = n$, then $T \vdash \varphi(\overline{m}, \overline{n})$,*
- ii. if $f(m) \neq n$, then $T \vdash \neg \varphi(\overline{m}, \overline{n})$.*

Now let’s introduce what is a variant idea of capturing, evidently closely related to that one, namely:

Defn. 54. *The theory T captures* the one-place function f by the open wff $\varphi^*(x, y)$ iff, for any m, n , if $f(m) = n$, then $T \vdash \forall y(\varphi^*(\overline{m}, y) \leftrightarrow y = \overline{n})$.*

It’s trivial that a wff that captures* f will also capture f (why?). And there’s nearly a converse:

Theorem 46. *If T extends Q , then if f is captured by some wff φ , then there’s also a wff φ^* which captures* f .*

There’s a little trick for defining a capturing* φ^* from a capturing φ , a trick which is explained in §12.2 of the book. But I’m not going to explain that here – it would be seriously boring to delay over the details. (And when I do a second edition of the book, I’m tempted to define capturing* from the off, and then we wouldn’t need the boring details!) So take the mini-theorem on trust, and work out how to prove it from the book if you must!

Now some more reminders:

Defn. 44. *The diagonalization of φ is $\exists y(y = \ulcorner \varphi \urcorner \wedge \varphi)$.*

Theorem 33. *There is a p.r. function $\text{diag}(n)$ which, when applied to a number n which is the g.n. of some wff, yields the g.n. of that wff’s diagonalization.*

And following on from those, here’s a new definition. If T is a theory that contains Q , it can capture (and hence capture*) all p.r. functions, then so in particular it can capture* the function diag . Hence we put

Defn. 55. *$\text{Diag}_T(x, y)$ is a T -wff which captures* diag .*

And now we can officially state and then prove

Theorem 47 (Diagonalization Lemma). *If T extends Q , and φ is a one-place open sentence of T ’s language, then there is sentence δ such that $T \vdash \delta \leftrightarrow \varphi(\ulcorner \delta \urcorner)$.*

Note by the way, T here doesn’t have to be a nicely axiomatized theory – just containing Q is enough.

To avoid unsightly rashes of subscripts, let’s henceforth drop subscript T s. Then we can argue like this:

Proof. Put $\alpha =_{\text{def}} \forall z(\text{Diag}(y, z) \rightarrow \varphi(z))$, and let δ be the diagonalization of α . Since diagonalizing α yields δ , we have $\text{diag}(\ulcorner \alpha \urcorner) = \ulcorner \delta \urcorner$. Hence (a) $T \vdash \forall z(\text{Diag}(\ulcorner \alpha \urcorner, z) \leftrightarrow z = \ulcorner \delta \urcorner)$ since by hypothesis Diag captures* diag in T . But just from the definition of δ it is equivalent to $\forall z(\text{Diag}(\ulcorner \alpha \urcorner, z) \rightarrow \varphi(z))$, and any theory containing a trivial amount of logic can prove that, so in particular (b) $T \vdash \delta \leftrightarrow \forall z(\text{Diag}(\ulcorner \alpha \urcorner, z) \rightarrow \varphi(z))$. Hence, substituting the provable equivalents from (a) into (b), we have $T \vdash \delta \leftrightarrow \forall z(z = \ulcorner \delta \urcorner \rightarrow \varphi(z))$, which trivially gives $T \vdash \delta \leftrightarrow \varphi(\ulcorner \delta \urcorner)$. \square

I promised that it was going to be easy!

Finally, a bit of jargon before proceeding. By a certain abuse of mathematical terminology, we say

Defn. 56. *If δ is such that $T \vdash \delta \leftrightarrow \varphi(\overline{\Gamma\delta})$, then it is said to be a fixed point for φ .*

So the Diagonalization Lemma is often called the Fixed Point Theorem – every one-place open sentence has a fixed point.

40 Incompleteness from the Diagonalization Lemma

You could skip this section at a first reading. What we do is first recover the First Incompleteness Theorem from the Diagonalization Lemma, and then fulfil our promise to show directly that $T \vdash G_T \leftrightarrow \neg \text{Prov}_T(\overline{\Gamma G_T})$.

40.1 Recovering the First Theorem

First we have the following general observation about provability predicates (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 \mathbf{Q} , and some sentence or other γ is a fixed point for $\neg \text{Prov}$ – i.e., $T \vdash \gamma \leftrightarrow \neg \text{Prov}(\overline{\Gamma\gamma})$. Then (i) if T is consistent, $T \not\vdash \gamma$. And (ii) if T is ω -consistent, $T \not\vdash \neg\gamma$.*

Proof. (i) Suppose $T \vdash \gamma$. Then $T \vdash \neg \text{Prov}(\overline{\Gamma\gamma})$. But if there is a proof of γ , then for some m , $\text{Prf}(m, \overline{\Gamma\gamma})$, so $T \vdash \text{Prf}(\overline{m}, \overline{\Gamma\gamma})$, since T captures Prf by Prf . Hence $T \vdash \exists x \text{Prf}(x, \overline{\Gamma\gamma})$, i.e. we also have $T \vdash \text{Prov}(\overline{\Gamma\gamma})$, making T inconsistent. So if T is consistent, $T \not\vdash \gamma$.

(ii) Suppose $T \vdash \neg\gamma$. Then $T \vdash \text{Prov}(\overline{\Gamma\gamma})$, i.e. $T \vdash \exists x \text{Prf}(x, \overline{\Gamma\gamma})$. But given T is consistent, there is no proof of γ , i.e. for every m , not- $\text{Prf}(m, \overline{\Gamma\gamma})$, whence for every m , $T \vdash \neg \text{Prf}(\overline{m}, \overline{\Gamma\gamma})$. So we have a φ such that T proves $\exists x \varphi(x)$ while it refutes each instance $\varphi(\overline{m})$, which makes T ω -inconsistent. So if T is ω -consistent, $T \not\vdash \neg\gamma$. \square

But now note that the general Diagonalization Lemma implies as a special case

Theorem 49. *There exists a sentence γ such that $T \vdash \gamma \leftrightarrow \neg \text{Prov}(\overline{\Gamma\gamma})$.*

Moreover, since Prov is Σ_1 , $\neg \text{Prov}$ is Π_1 , and the diagonalization construction produces a Π_1 fixed point γ . So putting those last two theorems together, we immediately recover Theorem 44.

40.2 Relating old and new

Briefly: how does the specific Gödel sentence G as we *originally* constructed it via the definitions in §33 stand to the generic Gödel sentences γ s we've just been talking about?

Well, how does our proof of the Diagonalization Lemma tell us to construct a γ such that $T \vdash \gamma \leftrightarrow \neg \text{Prov}(\overline{\Gamma\gamma})$? It says: first form a wff $\alpha = \forall z(\text{Diag}(y, z) \rightarrow \neg \text{Prov}(z))$, and then diagonalize α to get γ . So think more about α . Unpacking a bit, α is $\forall z(\text{Diag}(y, z) \rightarrow \neg \exists x \text{Prf}(x, z))$, which is equivalent to $\forall x \forall z \neg(\text{Diag}(y, z) \wedge \text{Prf}(x, z))$, i.e. to $\forall x \neg \exists z(\text{Diag}(y, z) \wedge \text{Prf}(x, z))$.

But now note that $\exists z(\text{Diag}(y, z) \wedge \text{Prf}(x, z))$ captures the *Gdl* relation. So this makes α pretty much like $\forall x \neg \text{Gdl}(x, y)$ – and hence γ (the diagonalization of α) is pretty like G (the diagonalization of $\forall x \neg \text{Gdl}(x, y)$).

40.3 Proving our old G_T is in fact a fixed point for $\neg \text{Prov}_T$

But let's now directly check that our old G_T is in fact a fixed point for $\neg \text{Prov}_T$, by giving a direct proof of

Theorem 50. *If T is p.r. axiomatized and contains \mathbf{Q} , $T \vdash G_T \leftrightarrow \neg \text{Prov}_T(\overline{\Gamma G_T})$.*

But don't get bogged down in this proof – I repeat it here from the book, just for the record.

Proof. Recall, dropping subscripts,

$$G =_{\text{def}} \exists y (y = \ulcorner U \urcorner \wedge U),$$

where ‘ $\ulcorner U \urcorner$ ’ stands in for the numeral for U ’s g.n.: further recall

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

where $\text{Gdl}(x, y)$ captures our old friend, the relation Gdl , where $Gdl(m, n)$ holds when m codes for a proof of the diagonalization of the wff with number n (a proof in T , of course!).

Now, by definition then,

$$Gdl(m, n) =_{\text{def}} \text{Prf}(m, \text{diag}(n)).$$

But the one-place p.r. function diag is captured* by an open wff $\text{Diag}(x, y)$. We can therefore now retrospectively fix on the following definition:

$$\text{Gdl}(x, y) =_{\text{def}} \exists z (\text{Prf}(x, z) \wedge \text{Diag}(y, z)).$$

And now let’s do some elementary manipulations:

$$\begin{aligned} G &\leftrightarrow \forall x \neg \text{Gdl}(x, \ulcorner U \urcorner) \\ &\leftrightarrow \forall x \neg \exists z (\text{Prf}(x, z) \wedge \text{Diag}(\ulcorner U \urcorner, z)) && \text{(definition of Gdl)} \\ &\leftrightarrow \forall x \forall z \neg (\text{Prf}(x, z) \wedge \text{Diag}(\ulcorner U \urcorner, z)) && \text{(pushing in the negation)} \\ &\leftrightarrow \forall z \forall x \neg (\text{Prf}(x, z) \wedge \text{Diag}(\ulcorner U \urcorner, z)) && \text{(swapping quantifiers)} \\ &\leftrightarrow \forall z (\text{Diag}(\ulcorner U \urcorner, z) \rightarrow \neg \exists x \text{Prf}(x, z)) && \text{(rearranging after ‘}\forall z\text{’)} \\ &\leftrightarrow \forall z (\text{Diag}(\ulcorner U \urcorner, z) \rightarrow \neg \exists v \text{Prf}(v, z)) && \text{(changing variables)} \\ &=_{\text{def}} \forall z (\text{Diag}(\ulcorner U \urcorner, z) \rightarrow \neg \text{Prov}(z)) && \text{(definition of Prov)} \end{aligned}$$

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

$$T \vdash G \leftrightarrow \forall z (\text{Diag}(\ulcorner U \urcorner, z) \rightarrow \neg \text{Prov}(z)).$$

Now, diagonalizing U yields G . Hence, just by the definition of diag , we have $\text{diag}(\ulcorner U \urcorner) = \ulcorner G \urcorner$. Since by hypothesis Diag captures* diag as a function, it follows by definition that

$$T \vdash \forall z (\text{Diag}(\ulcorner U \urcorner, z) \leftrightarrow z = \ulcorner G \urcorner).$$

Putting those two results together, we immediately get

$$T \vdash G \leftrightarrow \forall z (z = \ulcorner G \urcorner \rightarrow \neg \text{Prov}(z)).$$

But the right-hand side of that biconditional is trivially equivalent to $\neg \text{Prov}(\ulcorner G \urcorner)$. So we’ve proved the desired result. \square

41 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 two other peaks which can 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*.

41.1 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 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 imagine we have some expression $\mathsf{T}(x)$ which is suitably defined to express this numerical property $True$, and let L' be the result of adding $\mathsf{T}(x)$ to our initial language L . (For the moment, we leave it open whether L' is just L , which it would be if $\mathsf{T}(x)$ is in fact already definable from L 's resources.)

Then, for any L -sentence φ , we have

φ is true iff $True(\ulcorner\varphi\urcorner)$ iff $\mathsf{T}(\ulcorner\varphi\urcorner)$ is true.

Hence, for any L -sentence φ , every corresponding L' -sentence

$\mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$

is true. Which motivates our first main definition:

Defn. 57. An open L' -wff $\mathsf{T}(x)$ is a formal truth-predicate for L iff for every L -sentence φ , $\mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ is true.

And here's a companion definition:

Defn. 58. A theory Θ (with language L' which includes L) is a truth-theory for L iff for some L' -wff $\mathsf{T}(x)$, $\Theta \vdash \mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ for every L -sentence φ .

Equally often, a truth-theory for L is called a 'definition of truth for L '.

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.

41.2 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 $\mathsf{T}(x)$ such that $T \vdash \mathsf{T}(\ulcorner\varphi\urcorner) \leftrightarrow \varphi$ for every L -sentence φ . Since T has the right properties, the Diagonalization Lemma applies, so applying the Lemma to $\neg\mathsf{T}(x)$, we know that there must be some sentence L – a Liar sentence! – such that

1. $T \vdash \mathsf{L} \leftrightarrow \neg\mathsf{T}(\ulcorner\mathsf{L}\urcorner)$.

But, by our initial assumption, we also have

2. $T \vdash \mathsf{T}(\ulcorner\mathsf{L}\urcorner) \leftrightarrow \mathsf{L}$.

It is immediate that T is inconsistent, contrary to hypothesis. So our assumption must be wrong: T can't define truth for its own language. \square

41.3 The inexpressibility of truth

That first theorem puts limits on what a nice theory can *prove* about truth. But with very modest extra assumptions, we can put limits on what a theory's language can even *express* about truth.

Consider 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 Diagonalization Lemma applies, in particular to the negation of $\mathsf{T}_A(x)$. So we know that for some L_A sentence L ,

1. $\mathsf{Q} \vdash \mathsf{L} \leftrightarrow \neg\mathsf{T}_A(\ulcorner\mathsf{L}\urcorner)$.

But (and here comes the extra assumption we said we were going to invoke!) everything Q proves is true, since Q 's axioms are of course true and its logic is truth preserving. So

$$2. L \leftrightarrow \neg T_A(\overline{\neg L})$$

will also be a true L_A wff. But, by the assumption that T_A is a truth-predicate for L_A ,

$$3. T_A(\overline{\neg L}) \leftrightarrow L$$

must be true too. (2) and (3) immediately lead to contradiction again. Therefore our supposition that T_A is a truth-predicate has to be rejected. Hence no predicate of L_A can even express the numerical property $True_A$.

The argument evidently generalizes. Take any language L rich enough for us to be able to formulate in L something equivalent to the very elementary arithmetical theory Q (that's so we can prove the Diagonalization Lemma again). Call that an arithmetically adequate language. Then by the same argument, assuming Q is a correct theory,

Theorem 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 arithmetical truth) inside the theory.

41.4 A moral

Suppose T is a nice theory. Then (1) there are some numerical properties that T can capture (the p.r. ones for a start); (2) there are some properties that T can express but not capture (for example the property of Gödel-numbering a T -theorem – see the book, §21.4); and (3) there are some properties that T 's language L cannot even express (for example $True_L$, the numerical property of numbering-a-true- L -wff).

It is not, we should hasten to add, that the property $True_L$ is mysteriously ineffable, and escapes all formal treatment. A richer theory T' with a richer language L' may perfectly well be able to capture $True_L$. But the point remains that, however rich a given theory of arithmetic is, there will be limitations, not only on what numerical properties it can capture but even on which numerical properties that particular theory's language can express.

42 The Master Argument

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 , truth-in- L_A *isn't*. 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 with our emphasis back in §3 on 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 (think logical positivism, think Hilbertian instrumentalism), the very idea of truth-in-mathematics was under some suspicion. 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.

43 Rosser's Theorem

43.1 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 $T \not\vdash \neg G_T$.
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 $\neg R_T$ with a smaller code number).

43.2 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 $\overline{Prf}_T(x, y)$. So let's consider *the Rosser provability predicate* defined as follows:

Defn. 59. $RProv_T(x) =_{\text{def}} \exists v(\text{Prf}_T(v, x) \wedge (\forall w \leq v) \neg \overline{\text{Prf}}_T(w, x))$.

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 $T \vdash R_T \leftrightarrow \neg RProv_T(\ulcorner R_T \urcorner)$.*

We call such a sentence R_T a Rosser sentence for T .

Another semantic incompleteness result is immediate:

Theorem 54. *If T is a sound p.r. axiomatized theory including Q (and because sound therefore consistent), $T \not\vdash R_T$ and $T \not\vdash \neg R_T$, where R_T is a Rosser sentence*

Proof. Assume T 's soundness, then its theorems are true, and R_T is true if and only if it is not Rosser-provable. Suppose R_T were a theorem. Then it would be true since all theorems are true. So it is not Rosser-provable, which means that 'if R_T is provable, $\neg R_T$ is already provable' would be true, and also this conditional would have a true antecedent. We can infer that $\neg R_T$ is provable. Which makes T inconsistent, contrary to hypothesis. Therefore R_T is unprovable. Which shows that the material conditional 'if R_T is provable, $\neg R_T$ is already provable' has a false antecedent, and hence is true. In other words, R_T is true. Hence its negation $\neg R_T$ is false, and is therefore unprovable since only truths are provable in a sound theory. \square

As we said, however, in order to show that neither R_T nor $\neg R_T$ is provable we do not need the semantic assumption that T is sound. The syntactic assumption of T 's consistency is enough.

43.3 Rosser's Theorem

We can now show that

Theorem 55. *Let T be consistent p.r. axiomatized theory including Q and let ρ be any fixed point for $\neg R_{\text{Prov}_T}(x)$. Then $T \not\vdash \rho$ and $T \not\vdash \neg\rho$.*

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 55 is messy and very unpretty. Masochists can check out the proof of Theorem 21.2 in the book (on p. 178). 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 (that work is done on p. 179). You do not need to know these proofs! – just that they exist, so we get Rosser's Theorem (compare Theorem 45):

Theorem 56. *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 $T \not\vdash \varphi$ and $T \not\vdash \neg\varphi$.*

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 19–23, which I'm not going to be covering in lectures. Enthusiasts will want to devour the lot! – but let me especially highlight the sections which amplify this episode, and then the sections you ought to know about for general logical purposes anyway:

1. More on the topics in the episode: §§19.1–19.3, Ch. 20, §21.1–21.6: browse through Ch. 23.
2. For second-order arithmetics: §§22.1–22.6.

Gödel Without (Too Many) Tears – 10

Introducing the Second Theorem

Peter Smith

University of Cambridge

February 10, 2011

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

- 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

This episode introduces – at last – the *Second* Incompleteness Theorem, says just something about why it matters, and about what it takes to prove it.

Just two very quick reminders before we start. We said

Defn. 52. $\text{Prf}_T(x, y)$ stands in for a T -wff that canonically captures Prf_T ,

Defn. 53. Put $\text{Prov}_T(y) =_{\text{def}} \exists v \text{Prf}_T(v, y)$: such an expression is a provability predicate for T .

And then recall we proved:

Theorem 50. If T is p.r. axiomatized and contains \mathbf{Q} , $T \vdash \mathbf{G}_T \leftrightarrow \neg \text{Prov}_T(\overline{\mathbf{G}_T})$.

44 The Second Theorem introduced

44.1 Definitional preliminaries

We haven't put any requirement on the particular formulation of first-order logic built into \mathbf{Q} (and hence any theory which contains it). It may or may not have a built-in absurdity constant. But henceforth, let's use the sign ' \perp ' in the following way:

Defn. 60. ' \perp ' is T 's built-in absurdity constant if it has one, or else it is an abbreviation for ' $0 = \bar{1}$ '.

If T contains \mathbf{Q} , T of course proves $0 \neq \bar{1}$. So on either reading of ' \perp ', if T proves \perp , it is inconsistent. And if T 's logic is standard and it is inconsistent, then it will prove \perp .

Now consider the wff $\neg \text{Prov}_T(\overline{\perp})$. That is true if and only if T *doesn't* prove \perp , i.e. (given what we've just said) if and only if T is consistent. That evidently motivates the definition

Defn. 61. Con_T abbreviates $\neg \text{Prov}_T(\overline{\perp})$.

Note by the way that since Prov_T is Σ_1 , Con_T is Π_1 . 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 alternatives. Here's another pretty natural one. Suppose $\text{Contr}(x, y)$ 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. 62. $\text{Con}'_T =_{\text{def}} \neg\exists x\exists y(\text{Prov}_T(x) \wedge \text{Prov}_T(y) \wedge \text{Contr}(x, y))$.

But, on modest assumptions, this sort of definition and its variants are equivalent: so we'll stick to the crisp one.

44.2 The Formalized First Theorem

One half of the First Theorem tells us that, for suitable T (nicely axiomatized, containing \mathcal{Q}),

- (1) If T is consistent then \mathbf{G}_T is not provable in T .

(Remember, we only need the idea of ω -consistency for the other half of the First Theorem). Now, we can faithfully express (1) inside T itself by

- (2) $\text{Con}_T \rightarrow \neg\text{Prov}_T(\overline{\mathbf{G}_T})$.

But now reflect that the informal reasoning for the First Theorem is in fact rather elementary (we needed no higher mathematics at all, just simple reasoning about arithmetic matters). So we might well expect that if T contains enough arithmetic, it should itself be able to replicate that elementary reasoning.

In other words, if T is strong enough, then T can not only express (half) of the First Theorem via the wff abbreviated (2), but should be able to prove it too! – so we'd hope to have

Theorem 57. *For strong enough T , $T \vdash \text{Con}_T \rightarrow \neg\text{Prov}_T(\overline{\mathbf{G}_T})$.*

Call such a result the *Formalized First Theorem* for the relevant provability predicate. Suppose for now that we can indeed prove such a result.

44.3 The unprovability of consistency

We've just reminded ourselves of Theorem 50 which says that $T \vdash \mathbf{G}_T \leftrightarrow \neg\text{Prov}_T(\overline{\mathbf{G}_T})$. So putting that together with Theorem 57, we get

- (3) if $T \vdash \text{Con}_T$, then $T \vdash \mathbf{G}_T$.

But we know from the First Theorem that,

- (4) If T is consistent, $T \not\vdash \mathbf{G}_T$.

So the Formalized First Theorem immediately yields the unprovability of the relevant consistency sentence.

Theorem 58. *For strong enough T , if T is consistent, then $T \not\vdash \text{Con}_T$.*

Which is a somewhat vague version of the Second Incompleteness Theorem: roughly, for the right kind of theories T and the right kind of consistency sentences, T can't prove its own consistency sentences.

Obviously, we need to say something about what counts as a 'strong enough T '; but our vague statement will do as a very first introduction. Indeed, this is about as much as Gödel says in his original 1931 paper where he too didn't spell out the details.

45 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 \mathbf{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 59. *Suppose S is a consistent theory, strong enough for the Second Theorem to apply to it, and W is a fragment of S , then $W \not\vdash \text{Con}_S$.*

That's because, if S can't prove Con_S , a fortiori *part* of S can't prove it .

So, for example, we *can't* take some problematic rich theory like set theory which 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 §28.3 in these notes, and in the longer discussions in *IGT*).

46 What does it take to prove the Second Theorem?

46.1 Sharpening the Second Theorem

$\text{Prov}_T(y)$ abbreviates $\exists v \text{Prf}_T(v, y)$; 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 $T \vdash \text{Con}_T \rightarrow \neg \text{Prov}_T(\overline{\text{G}_T})$, if T 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. 63. *A theory is Σ -normal, if it is p.r. axiomatized, contains Q , and also also includes induction at least for Σ_1 wffs.*

Then the following looks a plausible conjecture

Theorem 60. *If T is Σ -normal, then T proves the formalized First Theorem, and so if T is consistent, $T \not\vdash \text{Con}_T$.*

(Warning: ' Σ -normal' is my shorthand: there seems to be no standard term here.) That sharpens our vaguely stated Theorem 58; and this better version is indeed provable. We won't give a full proof, however: but in the rest of the episode, we'll say something about how the details get filled in.

46.2 The box notation

To improve readability, let's introduce some notation. We will henceforth abbreviate $\text{Prov}_T(\overline{\varphi})$ simply by $\Box_T \varphi$. 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, $\neg \text{Prov}_T(\overline{\text{G}_T})$ can be abbreviated $\neg \Box_T \text{G}_T$. Thus in our new notation, the Formalized First Theorem is $T \vdash \text{Con}_T \rightarrow \neg \Box_T \text{G}_T$. Moreover, Con_T can now alternatively be abbreviated as $\neg \Box_T \perp$.

However, we will very often drop the explicit subscript from the box symbol and elsewhere, and let context supply it.

46.3 The ‘Derivability Conditions’ and a tiny amount of history

First a standard definition: we will say (dropping subscripts)

Defn. 64. *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 61. *If T is Σ -normal, then the derivability conditions hold for T .*

Theorem 62. *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 60, so we can concentrate on them. Now, proving Theorem 61 in detail is a seriously tedious task, and I don't propose to do it here, nor do I do it in the current edition of *IGT* either – though I do outline proof-sketches there in §§26.1–26.3. But we *will* here prove Theorem 62.

But first, a *very* small amount of history. As we remarked before, Gödel himself didn't prove the Formalized First Theorem for his particular formal theory P . The hard work was first done for a different theory by David Hilbert and Paul Bernays in their *Grundlagen der Mathematik* of 1939: the details of their proof are in fact 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 definition above. The derivability conditions are consequently sometimes called the HBL conditions.

46.4 Deriving the Formalized First Theorem

As announced, we'll now prove Theorem 62. We assume T is Σ -normal (in fact, all we need is that it is p.r. axiomatized and contains \mathbf{Q}), and that the derivability conditions hold, and aim to show (dropping subscript ' T 's) $T \vdash \text{Con} \rightarrow \neg\Box\mathbf{G}$.

Proof. First, since T is p.r. axiomatized and contains \mathbf{Q} , Theorem 50 holds. So, in our new symbolism, we have $T \vdash \mathbf{G} \leftrightarrow \neg\Box\mathbf{G}$.

Second, note that for any theory T containing \mathbf{Q} , $T \vdash \neg\perp$ (either by the built-in logic, or because we've put $\perp =_{\text{def}} \mathbf{0} = \bar{1}$). And simple logic will show that, for any wff φ , we have

$$T \vdash \neg\varphi \rightarrow (\varphi \rightarrow \perp).$$

Given the latter and the derivability condition (C1), this means

$$T \vdash \Box(\neg\varphi \rightarrow (\varphi \rightarrow \perp)).$$

So given the derivability condition (C2) and using modus ponens, it follows that for any φ

$$\text{A. } T \vdash \Box\neg\varphi \rightarrow \Box(\varphi \rightarrow \perp).$$

We now argue as follows:

- | | |
|---|-------------------|
| 1. $T \vdash \mathbf{G} \rightarrow \neg\Box\mathbf{G}$ | Half of Thm 50 |
| 2. $T \vdash \Box(\mathbf{G} \rightarrow \neg\Box\mathbf{G})$ | From 1, given C1 |
| 3. $T \vdash \Box\mathbf{G} \rightarrow \Box\neg\Box\mathbf{G}$ | From 2, given C2 |
| 4. $T \vdash \Box\neg\Box\mathbf{G} \rightarrow \Box(\Box\mathbf{G} \rightarrow \perp)$ | Instance of A |
| 5. $T \vdash \Box\mathbf{G} \rightarrow \Box(\Box\mathbf{G} \rightarrow \perp)$ | From 3 and 4 |
| 6. $T \vdash \Box\mathbf{G} \rightarrow (\Box\Box\mathbf{G} \rightarrow \Box\perp)$ | From 5, given C2 |
| 7. $T \vdash \Box\mathbf{G} \rightarrow \Box\Box\mathbf{G}$ | Instance of C3 |
| 8. $T \vdash \Box\mathbf{G} \rightarrow \Box\perp$ | From 6 and 7 |
| 9. $T \vdash \neg\Box\perp \rightarrow \neg\Box\mathbf{G}$ | Contraposing |
| 10. $T \vdash \text{Con} \rightarrow \neg\Box\mathbf{G}$ | Definition of Con |

Which gives us the Formalized First Theorem (and hence, as before, the Second Incompleteness Theorem). □

Curry's Paradox, Löb's Theorem and other excitements

Peter Smith

University of Cambridge

February 10, 2011

PDFs of this series of notes are published on my website www.logicmatters.net and then escape into the wild to roam uncontrolled! Visit the site to get the latest version, and perhaps offer comments, etc.

- Curry's Paradox
 - Löb's Theorem
 - Löb's Theorem implies the Second Incompleteness Theorem again
 - Con_T as an undecidable sentence
 - Consistent theories that 'prove' their own inconsistency
 - What's still to come ...
-

This episode starts by introducing Curry's Paradox, and we then use the simple core reasoning that leads to the paradox to prove a theorem, which we can call Curry's 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). *Very* roughly then, we could say that as Gödel's First Incompleteness Theorem stands to the Liar Paradox, Gödel's Second Theorem stands to Curry's Paradox.

We then note some additional facts about consistency sentences, in particular remarking that there are consistent theories that 'prove' their own inconsistency (how can that be? – because they are ω -inconsistent and so are unreliable theories which tell fibs about the natural numbers!).

We finish with a quick overview of where the rest of *IGT* goes, but – at least in these lectures – we can't follow the story any further here.

47 From Curry to Löb

47.1 How to prove that the moon is made of green cheese

We seemingly can construct a sentence that in effect says of itself e.g. 'if I am true, then the moon is made of green cheese (after all, I've apparently just constructed one!). Let's replace the indexical 'I' with a suitable definite description, and then symbolize that 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 . Assuming the usual equivalence between δ and δ 's being true – an assumption we'll revisit – then we have $\delta \leftrightarrow (\delta \rightarrow \varphi)$. 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 tidy up 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 propositions like (1) seem to be available to us, given we can construct self-referential sentences (surely often harmless, as in 'This sentence contains five words') and we have a truth-predicate for which, intuitively, δ is true holds if and only if δ . 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. True, we do use the supposition at line (2) and also the conditional at line (6) twice; but why should *that* matter?¹

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 doesn't explicitly involve negation. So even if boxing clever with negation might provide a response to the Liar Paradox (we might go paraconsistent, 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 such an approach to the Liar.

47.2 Curry's theorem, and two implications

Let's now go more formal, and turn the informal reasoning for a paradox into formal reasoning for a theorem.

So, we will say that a theory T – take that as usual to be a properly formalized theory – *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 have evidently have the following formal result we can call Curry's Theorem:²

Theorem 63. *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.*

¹OK, 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.

²I'd better remark that this is *not* a standard label.

For if the condition of this theorem is satisfied, then for all φ , T proves some $\delta \leftrightarrow (\delta \rightarrow \varphi)$ and hence – by the reasoning given above – T proves φ . And proving every sentence φ , T is inconsistent.

Let’s draw two conclusions from this abstract but simple result. Both are closely related to things we already know. Still, it is always nice and can often be illuminating to get at variants of old conclusions by new routes.

So first let’s apply Curry’s Theorem to set theory. The naive comprehension theorem, recall, says that

For any monadic condition $\chi(x)$, $\exists_1 y \forall x (x \in y \leftrightarrow \chi(x))$.

We already know that naive comprehension leads to trouble via Russell’s Paradox in a theory equipped with a sensible negation. But Curry’s Theorem shows that naive comprehension still leads to trouble however we doctor negation so long as the conditional remains standard:

Theorem 64. *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 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 are done; we’ve shown that for any φ you choose, there’s a δ such that T proves $\delta \leftrightarrow (\delta \rightarrow \varphi)$. So such a T is inconsistent by Curry’s Theorem, contrary to hypothesis. \square

Our next application takes us back to issues about truth again. Suppose theory T has enough arithmetic to prove a Diagonalization Lemma, so that for every one-place open sentence $\varphi(x)$ of T ’s language, there is some fixed point sentence δ such that $T \vdash \delta \leftrightarrow \varphi(\ulcorner \delta \urcorner)$ where $\ulcorner \delta \urcorner$ is T ’s numeral for $\ulcorner \delta \urcorner$, the Gödel number of δ on some sensible coding scheme. Let’s say that such a theory ‘knows some arithmetic’ (because, of course, even containing the very weak Robinson arithmetic is enough for T to prove the Diagonalization Lemma).

Let’s also say that T ‘has a naive theory of truth’ if we can form an open sentence $\top(x)$ in T such that $T \vdash \top(\ulcorner \delta \urcorner) \leftrightarrow \delta$ unrestrictedly, for every sentence δ (i.e. all ‘ \top -biconditionals’ are provable). Then we have a version of Tarski’s theorem (compare Theorem 51):

Theorem 65. *If T is consistent, has a standard conditional and knows some arithmetic, 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 $\top(x) \rightarrow \varphi$. The Diagonalization Lemma tells us that for some δ , T proves $\delta \leftrightarrow (\top(\ulcorner \delta \urcorner) \rightarrow \varphi)$. By the assumption that $T \vdash \top(\ulcorner \delta \urcorner) \leftrightarrow \delta$, it follows that T proves $\delta \leftrightarrow (\delta \rightarrow \varphi)$. So by Curry’s Theorem, T would be inconsistent. \square

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. 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’.”

47.3 Proving the moon is made of green cheese, again

Let’s now, for brevity’s sake, rewrite ‘ $\top(\ulcorner \delta \urcorner)$ ’ as ‘ $\top\delta$ ’. (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. We’ll use the different type face for ‘ \top ’ to remind ourselves that it isn’t itself the truth-predicate \top but an abbreviation for a complex expression involving the truth-predicate.)

Then the derivation of Theorem 3 depends on the assumption that (for any δ) $T \vdash \top\delta \leftrightarrow \delta$. But of course for other ‘modalities’ we don’t have this kind of general ‘modal collapse’. So we won’t be able directly to transpose the reasoning so far to give results about other cases.

But in fact, we don't need the full content of the 'naive theory of truth' to get a version of Curry's paradox; and this new version *will* carry over to other 'modalities'.

To explain, suppose we are dealing with a \top -predicate in a theory T for which the following principles hold (schematically in φ, ψ):

- T0. $T \vdash \top\varphi \rightarrow \varphi$,
- T1. $T \vdash \top(\varphi \rightarrow \psi) \rightarrow (\top\varphi \rightarrow \top\psi)$,
- T2. $T \vdash \top\varphi \rightarrow \top\top\varphi$,
- T3. If $T \vdash \varphi$ then $T \vdash \top\varphi$.

These principles are parallel to those that hold for the modality S4, and we know that modality in S4 doesn't collapse, so we know that these principles don't entail that T has a full naive truth-theory and can always prove $\top\varphi \leftrightarrow \varphi$.

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 S4-ish principles we've just given.

| | | |
|----|---|--|
| 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) T0 to T3. The improved version of Theorem 65 is

Theorem 66. *If T is consistent, has a standard conditional and knows some arithmetic, it can't also have a truth-predicate \top for which conditions T0 to T3 hold.*

Proof. Assume for reductio that T has a truth-predicate for which conditions T0 to T3 hold. Take any φ and form the predicate $\top(x) \rightarrow \varphi$. Since T knows some arithmetic, so proves the Diagonalization Lemma, there is some δ such that $T \vdash \delta \leftrightarrow (\top(\overline{\delta}) \rightarrow \varphi)$, i.e. in shorthand $T \vdash \delta \leftrightarrow (\top\delta \rightarrow \varphi)$. The proof then continues as above to derive φ . Since φ was arbitrary, that makes T inconsistent, contrary to hypothesis. \square

47.4 Löb's Theorem

Now, conditions $\top 1$ to $\top 3$ on the 'modality' \top (defined from the truth-predicate $\top(x)$) exactly match, numbering apart, the so-called derivability conditions on \Box (which we defined from the provability-predicate $\text{Prov}(x)$). Here are those conditions again:

Defn. 64. *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 $(\text{Prov}(z) \rightarrow \varphi)$. 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 $\delta \leftrightarrow (\text{Prov}(\ulcorner \delta \urcorner) \rightarrow \varphi)$, or for short, $T \vdash \delta \leftrightarrow (\Box\delta \rightarrow \varphi)$. Then, at 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 the following theorem:

Theorem 67 (Löb). *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 .

47.5 On a question of Henkin's

By the Diagonalization Lemma applied to the unnegated wff $\text{Prov}(x)$, there is a sentence H such that $T \vdash H \leftrightarrow \text{Prov}(\ulcorner H \urcorner)$ – 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?*

It is. For by hypothesis, $T \vdash \text{Prov}(\ulcorner H \urcorner) \rightarrow H$, i.e. $T \vdash \Box H \rightarrow H$; so $T \vdash H$ by Löb's Theorem.

47.6 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 $T \vdash \Box\perp \rightarrow \perp$ then $T \vdash \perp$. Hence, if $T \not\vdash \perp$, so T is consistent, then $T \not\vdash \Box\perp \rightarrow \perp$, hence $T \not\vdash \neg\Box\perp$ hence (just by definition of Con), $T \not\vdash \text{Con}$.

There's a converse too, i.e. the Second Theorem quickly entails Löb's Theorem (*IGT*, p. 231). 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, Liar-style reasoning can lead not to paradox but a 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.

48 Other excitements

We'll assume throughout this section that T is such that the derivability conditions hold (Σ -normality suffices, see Def. 63).

48.1 G_T and Con_T are provably equivalent in T

We know from Theorem 62 that $T \vdash \text{Con} \rightarrow \neg \Box G$ (suppressing subscripts!).

Now note the following result which says that T knows that, *if* it can't prove φ , it must be consistent.

Theorem 68. *For any sentence φ , $T \vdash \neg \Box \varphi \rightarrow \text{Con}$.*

Proof. We argue as follows:

- | | |
|---|-------------------|
| 1. $T \vdash \perp \rightarrow \varphi$ | Logic! |
| 2. $T \vdash \Box(\perp \rightarrow \varphi)$ | From 1, given C1 |
| 3. $T \vdash \Box \perp \rightarrow \Box \varphi$ | From 2, given C2 |
| 4. $T \vdash \neg \Box \varphi \rightarrow \neg \Box \perp$ | Contraposing |
| 5. $T \vdash \neg \Box \varphi \rightarrow \text{Con}$ | 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. \square

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 $T \vdash \neg \Box G \rightarrow \text{Con}$. So putting that together with Theorem 62, we have $T \vdash \text{Con} \leftrightarrow \neg \Box G$. And now combine *that* with Theorem 50 which tells us that $T \vdash G \leftrightarrow \neg \Box G$, and low and behold we've shown

Theorem 69. *If T is Σ -normal then $T \vdash \text{Con} \leftrightarrow G$.*

This means that, not only do we have $T \not\vdash \text{Con}$, we also have (assuming T is ω -consistent) $T \not\vdash \neg \text{Con}$. 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.

48.2 Theories that 'prove' their own inconsistency

An ω -consistent T can't prove $\neg \text{Con}_T$, as we've just noted. By contrast, a consistent but ω -inconsistent T might well have $\neg \text{Con}_T$ 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 70. *Under the given conditions, $\vdash \neg \text{Con}_S \rightarrow \neg \text{Con}_R$.*

Proof. Suppose $\neg \text{Con}_S$, i.e. suppose $\exists v \text{Prf}_S(v, \perp)$. Hence for some \mathbf{a} , $\text{Prf}_S(\mathbf{a}, \perp)$. And that implies $\text{Prf}_R(\mathbf{a}, \perp)$. 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 $\exists v \text{Prf}_R(v, \perp)$, i.e. $\neg \text{Con}_R$. And the inferences here only use first-order logic. \square

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 $\neg \text{Con}_{\text{PA}}$.

By definition, $R \vdash \neg \text{Con}_{\text{PA}}$. So using our lemma we can conclude $R \vdash \neg \text{Con}_R$. R is ω -inconsistent (why? because that means $R \vdash \neg G_R$, 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 $\text{PA} \vdash \text{Con}_{\text{PA}}$, then PA would be inconsistent). So,

Theorem 71. *Assuming PA is consistent, the theory $R = \text{PA} + \neg \text{Con}_{\text{PA}}$ is a consistent theory which 'proves' its own inconsistency.*

And since R proves $\neg\text{Con}_R$,

Theorem 72. *The consistent theory R is such that $R + \text{Con}_R$ 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 $\neg\text{Con}_{\text{PA}}$. 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 69, $R \vdash \text{Con}_R \leftrightarrow G_R$. So since $R \vdash \neg\text{Con}_R$, $R \vdash \neg G_R$. 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 .

There's lots more in the book around and about these issues, in Chaps 24–28. If you are going to read more, however, perhaps the three things to concentrate on are these. First, the discussion of Hilbert's Programme in the first half of Chap. 28 (as the topic is historically important). Second, the quick remarks about minds and machines later in Chap. 28. And then – for enthusiasts – there are the more intricate arguments in Chap. 27 leading up to discussion of what Gödel called the 'best and more general version' of the Second Theorem.

49 What's still to come . . .

And here, my lectures this year have to end. But the book carries on. So let me *very* 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 23.)

Compare 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*. 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 \mathbf{Q} , and hence \mathbf{PA} , can capture all the p.r. functions: we next show that they can capture all the μ -recursive functions. The fact that \mathbf{Q} and \mathbf{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 . We show that the unsolvability of the halting problem gives us another proof that it isn't mechanically decidable what's a theorem of first-order logic, and it also entails Gödelian incompleteness again.
5. The fact that two independent ways of trying to characterize the class of computable functions coincide supports what we can now call the *Church-Turing Thesis*, which underlies the links we need to make e.g. between formal results about what a Turing machine can decide and results about what is effectively decidable in the intuitive sense. We finish the book by discussing the Church–Turing Thesis further, and consider its status.