Chapter 10

Gödel's Demon

I will now argue that Gödel's second incompleteness theorem is another example of the sort of skeptical, demonic argument that shows that $\neg K$ is not formally verifiable.

10.1 Computing is Proving

Consider the first-order language L of arithmetic, which includes the vocabulary:

1. $\forall, \vee, \neg,$
2. $s, +, \cdot,$
3. $=$
4. $\mathbf{0}$.

Let $\vdash$ be a sound, complete first-order proof system for the preceding language. I'll choose natural deduction. I won't be so fussy about providing every formal detail, since you have done this before and know what the details are like. Consider the following, very weak axioms called Q (another lovely mnemonic).

1. $(\forall x)(\forall y)\ (s(x) = s(y) \rightarrow x = y)$;
2. $(\forall x)\ (\mathbf{0} \neq s(x))$;
3. $(\forall x)(x \neq \mathbf{0} \rightarrow (\exists y)\ (x = s(y)))$
4. $(\forall x)(x + \mathbf{0} = x)$;
5. $(\forall x)(\forall y)\ (x + s(y) = s(x + y))$;
6. $(\forall x)\ (x \cdot \mathbf{0} = 0)$;
7. $(\forall x)(\forall y)\ (x \cdot s(y) = (x \cdot y) + x)$.

Axioms 1-3 provide formal control over the successor function, axioms 3, 4 are the usual primitive recursive definition of + and axioms 5, 6 are the primitive recursive definition of $\cdot$. Think of Q as a really bare-bones theory of arithmetic. For example, you can prove that 2+2 = 4.

1.	$s(s(\mathbf{0}))+\mathbf{0}$	$=$	$s(s(\mathbf{0}))$	ax 4;
2.	$s(s(\mathbf{0}))+s(\mathbf{0})$	$=$	$s[s(s(\mathbf{0}))+\mathbf{0}]$	ax 5;
3.	$s(s(\mathbf{0}))+s(\mathbf{0})$	$=$	$s(s(s(\mathbf{0})))$	1, 2, =
4.	$s(s(\mathbf{0}))+s(s(\mathbf{0}))$	$=$	$s[s(s(\mathbf{0}))+s(\mathbf{0})]$	ax 5;
5.	$s(s(\mathbf{0}))+s(s(\mathbf{0}))$	$=$	$s(s(s(s(\mathbf{0}))))$	3, 4, =.

Notice that this derivation witnessing $Q \vdash 2+2=4$ may be viewed not only as a proof but as a *computation* of $2+2$, since the derivation builds up the value of $2+2$ in the obvious, primitive recursive way from more basic values. The same could be said of multiplication.

That suggests an intriguing idea: although you tend to think of proofs as *justifications of claims*, they can also be viewed as *computations of functions* in the following way. Say that $\Phi(\vec{x},y)$ *represents* f in system $T \supseteq Q$ iff for each n,m,

$$f(\vec{n})=m \quad \Rightarrow \quad T \vdash \Phi(\vec{\mathbf{n}},\mathbf{m}) \wedge (\forall z)\ (\Phi(\vec{\mathbf{n}},z) \rightarrow z=\mathbf{m}).$$

Think of "Q represents" as nothing different than "Q computes". If proofs in Q are thought of as computations, then it is best to conceive of Q as a "non-deterministic" computing formalism. Non-determinism allows for alternative computational paths once the device is started. For example, Turing machines can be equipped with indeterministic transitions. Then it is possible for two different computational paths to yield both $\Phi(x,y)$ and $\Phi(x,y')$, where $y \neq y'$. Note that representability is defined in terms of entailment, rather than non-entailment, so that if Q entails everything, then every possible function is representable. That's not a good thing, for it amounts to saying that Q might non-deterministically return any value when asked to produce $f(n)$. If you ask Q if it does that, it will of course say "no". But it will also say "yes"! We are interested in the case in which Q is consistent, rather than in these Zen pronouncements.

Here are some examples of prominent functions and the formulas that represent them in Q:

$$\begin{aligned}
+ &: x+y=z,\\
\cdot &: x \cdot y=z,\\
o &: x=x \wedge y=\mathbf{0},\\
p_i^n &: x_0=x_0 \wedge \ldots \wedge x_n=x_n \wedge y=x_i,\\
\psi \circ \gamma &: (\exists z)\ (\Gamma(x,z) \wedge \Psi(z,y),\\
(\mu z)\psi &: (\exists z)\ (\Psi(x,z) \wedge (\forall w)\ (w<z \rightarrow \neg\Psi(x,z))).
\end{aligned}$$

In the last formula, the strict inequality may be eliminated by the definition: $w<z \leftrightarrow (\exists y)\ (s(y)+w=z)$. Notice that the only unbounded quantifiers occurring in these formulas are existential. It turns out that

Proposition 10.1 (Arithmetical representation) *Every total recursive function is representable in Q (by a formula whose unbounded quantifiers are all existential).*

I'm not going to detain you with the messy coding details and all the little proofs in Q that are required to show this. See (Boolos and Jeffrey) for the full argument. The basic message is that *computing is a kind of proving.* You may ask why Gödel doesn't argue that all partial recursive functions are representable. That's because representability of partial functions would involve a non-entailment condition when the function is undefined. Representability of total functions can be established entirely in terms of concrete proofs in Q, without recourse to non-entailment claims, which involve infinite model constructions that Gödel's finitistic audience would have rejected.

You may now think of Q as deciding predicate P, just in case Q computes the characteristic function of P. Alternatively, say that formula Φ *represents* P iff

$$\begin{aligned} P(n_0,\ldots,n_k) &\Rightarrow Q \vdash \Psi(\mathbf{n_0},\ldots,\mathbf{n_k});\\ \neg P(n_0,\ldots,n_k) &\Rightarrow Q \vdash \neg\Psi(\mathbf{n_0},\ldots,\mathbf{n_k}) \end{aligned}$$

Proposition 10.2 (Arithmetical representation theorem) *Every recursive relation $R(x_1,\ldots,x_n)$ is represented in Q by some open formula $\Phi^{x_1,\ldots,x_n}$ in which all unbounded quantifiers are outside of negations and are existential.*

When you represent recursive relation R with Φ in this way, it appears that there is no "output" of "program" Ψ. But if you view proofs in P as computations, then there *is* a tacit output to the computation $\Psi(n)$, namely, the Boolean value indicated by whether $\Psi(\mathbf{n})$ occurs negated or non-negated in the proof. If Q is consistent, then there is no way that one can obtain both answers on the same inputs, in spite of the indeterminism of computations in Q. Otherwise, Q eventually produces both answers by potentially distinct pathways. Notice, that Q needn't come to both of these incompatible conclusions simultaneously. That's what makes consistency interesting— an inconsistent theory may empirically "seem" to be consistent and informative for an arbitrary amount of time before any explicit contradiction appears.

10.2 Proving is Computing

Think of an open formula $\Phi^k(x_1,\ldots,x_n)$ as a "program" for deciding some relation $R(x_1,\ldots,x_n)$in "programming language Q". Effectively assign Gödel numbers to open formulas of L free only in variable x whose unbounded quantifiers are all non-negated and existential, so that you have $\Phi^1_0(x), \Phi^1_1(x),\ldots$. Also, effectively assign numbers to proofs in Q in some obvious way (each proof is a sequence of code-able things). Although formal proofs in system Q are hard to find, they are trivial to check because all the steps are explicitly justified in a

mechanical way, as long as it is effectively decidable what counts as an axiom. Define

$$Proof_Q(i,t) \iff t \text{ codes a proof in } Q \text{ of the statement coded by } i.$$

Then (by the Church-Turing thesis and the formal rules of proof checking in Q), Proof_Q is recursive.

Also it's easy to decode a formula, instantiate a constant, and then re-code the result, so by the Church-Turing thesis, there is a total recursive s such that for each n,

$$\Phi^0_{s(i,n)} = \Phi^1_i(\mathbf{n}).$$

Then the following relation is recursive:

$$Proof_Q(i,t,n) \iff Proof_Q(f(i,n),t).$$

In other words,

$$Proof_Q(i,t,n) \iff t \text{ codes a proof in } Q \text{ of } \Phi(\mathbf{n}), \text{ where } i \text{ codes } \Phi(x).$$

If you think about Q as a programming system, $Proof_Q(i,t,n)$ is analogous to the universal relation $U(i,t,1,\langle n\rangle)$, since the proof t is a witness of computation length and n is the "input" and the tacit output is unity (with respect to the formula coded by i, since it is proved rather than its negation and negation determines the tacit output of the computation.

10.3 The Gödel Sentence as Demonic Strategy Against Q

Recall the skeptical argument that $\neg K$ is not r.e. We used the Kleene recursion theorem to construct a "demon" index d that watches what your favorite method would do on input d and then halts as soon as your method becomes sure that it won't. Gödel's first incompleteness theorem can be viewed in much the same way, as a Humean skeptical argument against a proposed proof system. Proof systems aren't omniscient; they have to chug along in the manner described above, and a demonic statement (program) can "watch" these proceedeings. Indeed, it can "make itself false" as soon as it "sees" that the system has concluded it is true, just the way our demonic index returns 0 on each input as soon as your method is sure that it is undefined on its own index.

So how does the computation directed by a sentence Φ^0_d "return zero" as soon as Q concludes it is true by proving it? By directing a computation that "simulates" Q trying to prove it and that returns zero (itself negated) when that happens! By the representation theorem, let formula $\Psi(i,t,n)$ represent $Proof_Q(i,t,n)$ in Q. Consider the formula

$$(\forall t)\ \neg\Psi(i,t,i).$$

As a formula free just in i, this formula has a Gödel number d, so $\Phi^1_d(i) = (\forall t)\ \neg\Psi(i,t,i)$. This "says" of i that $\Phi^1_i(\mathbf{i})$ is not provable in Q. Notice how analogous this is to saying $\phi_i(i)\uparrow$, which amounts to $i \in \neg K$, since finding a proof in Q that $\Phi_i(\mathbf{i})$ is like $\phi_i(i)$ halting.

Now let

$$D = (\forall t)\ \neg\Psi(\mathbf{d},t,\mathbf{d}).$$

This is the celebrated Gödel sentence. Notice that D just is the result of substituting $\mathbf{d}$ into $\Phi_d(i)$, so this sentence "says" of itself that it is not provable. Now suppose that

$$Q \vdash D.$$

Then for some t,

$$Proof_Q(d,t',d).$$

So since Ψ represents $Proof_Q$,

$$Q \vdash \Psi(\mathbf{d},\mathbf{t},\mathbf{d}).$$

So by the proof theoretical rules governing introduction of $\exists$,

$$Q \vdash (\exists t)\ \Psi(\mathbf{d},t,\mathbf{d}).$$

Since DeMorgan's rules hold for first-order logic, it follows that

$$Q \vdash \neg(\forall t)\ \neg\Psi(\mathbf{d},t,\mathbf{d}).$$

So

$$Q \vdash \neg D.$$

So Q is inconsistent, contrary to assumption. By *reductio ad absurdum*, $Q \nvdash D$.

The argument so far corresponds to what happens if D sees Q become sure that Q won't ever become sure of D. In that case, D directs a non-deterministic computation in which Q becomes sure of D, so Q is wrong. But given the special nature of computation in Q, this amounts to an outright inconsistency in Q. I view that as incidental. The real story is that D "out-waits" Q and makes what Q did wrong.

If D is a good demon, D should also embarrass Q if Q stubbornly refuses to become sure that Q will never become sure of D. But that is just what happens. For suppose that

$$Q \nvdash D,$$

so for each t,

$$\neg Proof_Q(d,t,d).$$

So since Ψ represents $Proof_Q$, you have for each t,

$$Q \vdash \neg\Psi(\mathbf{d},\mathbf{t},\mathbf{d}).$$

Now suppose for *reductio* that nonetheless

$$Q \vdash \neg D,$$

so

$$Q \vdash (\exists t)\ \Psi(\mathbf{d}, t, \mathbf{d}).$$

Thus, Q says that something satisfies Ψ, but says that no particular natural number satisfies Ψ. That isn't inconsistent, because Q doesn't say that everything is a natural number (how could it?). But if Q could say that everything is a natural number, it would be inconsistent. Gödel calls this situation *omega-inconsistency*. Hence we have:

Proposition 10.3 (Gödel's second incompleteness theorem)

If Q is consistent, then $Q \nvdash D$;

if Q is omega-consistent, then $Q \nvdash \neg D$.

10.4 The Wrong View

One is tempted to say that "D says (timelessly) that it is unprovable in Q" or that you can "see that D is true". Both statements are misleading. In general, Gödel's audience for his incompleteness theorems consisted of finitists and formalists, who were deeply suspicious of talk of univocal "truth" in mathematics. Therefore, Gödel went out of his way to exclude "truth" and "meaning" from the statement or the proof of his incompleteness theorem. To say that you can "see" the truth of the Gödel sentence is even worse, since you can only "see" that it is true *if* you can "see" that Q is consistent. For if Q is inconsistent, every sentence is provable, including D, which supposedly "truly" says "I am not provable". But I don't think you can "see" that Q is consistent. You simply believe it on the basis of long experience devoid of contradictions, the way scientists believe their theories until something goes explicitly wrong.

That's my response to Penrose and Lucas, who thought Gödel showed that humans can't be computers. "Seeing" consistency is not infallible verification by means of an uncomputable crystal ball in the mind that guarantees no surprises will be faced in the future. It is just fallible, empirical guesswork that will disappear in a flash when something does go wrong. And Turing machines can act confident until something explicitly goes wrong in an effective simulation just as well as we can!