Computing the Uncomputable

Re-written for clarity by Florian Suess

John Carlos Baez

August 26, 2022

I love the more mind-blowing results of mathematical logic for example from **Joel David Hamkins**; "Any function can be computable."

Let me try to explain it without assuming you're an expert on mathematical logic. That may be hard, but I'll give it a try. We need to start with some background.

Preliminaries

First, you need to know that there are many different 'models' of arithmetic. If you write down the usual axioms for the natural numbers, the Peano axioms (or 'PA' for short), you can then look around for different structures that obey these axioms. **These are called 'models' of PA**.

One of them is what you think the natural numbers are. For you, the natural numbers are just $0, 1, 2, 3, ...$, with the usual way of adding and multiplying them. This is usually called the 'standard model' of PA. The numbers $0, 1, 2, 3, ...$ are called the 'standard' natural numbers.

Detour

But there are also **nonstandard models** of arithmetic. These models contain extra numbers beside the standard ones! These are called 'nonstandard' natural numbers. *This takes a while to get used to. There are several layers of understanding to pass through.*

For starters, you should think of these extra 'nonstandard' natural numbers as bigger than all the standard ones. So, imagine a whole bunch of extra numbers tacked on after the standard natural numbers, with the operations of addition and multiplication cleverly defined in such a way that all the usual axioms still hold.

You can't just *tack on finitely many extra numbers* and get this to work. But there can be countably many, or uncountably many. There are infinitely many different ways to do this. They are all rather hard to describe.

Perspective

To get a handle on them, it helps to realize this. Suppose you have a statement S in arithmetic that is neither provable nor disprovable from PA. Then S will hold in some models of arithmetic, while its negation S will hold in some other **models**.

For example, the Gödel sentence G says "this sentence is not provable in PA". If Peano arithmetic is consistent, neither G nor G is provable in PA. So G holds in some models, while G holds in others.

Thus, you can intuitively think of different models as "possible worlds". If you have an undecidable statement, meaning one that you can't prove or disprove in PA, then it holds in some worlds, while its negation holds in other worlds.

In the case of the Gödel sentence G, most mathematicians think G is "true". Why the quotes? Truth is a slippery concept in logic... as there is no precise definition of what it means for a sentence in arithmetic to be "true". All we can precisely define is:

1. whether or not a sentence is provable from some axioms, and,

2. whether or not a sentence holds in some model.

Nonetheless, mathematicians are human, so they have beliefs about what's true. Many mathematicians believe that G is true: indeed, in popular accounts one often hears that G is "true but unprovable in Peano arithmetic". So, these mathematicians are inclined to say that any model where G doesn't hold is nonstandard.

The Result

Anyway, what is Joel David Hamkins' result? It's this:

There is a Turing machine T with the following property. For any function f from the natural numbers to the natural numbers, there is a model of PA such that in this model, if we give T any standard natural n as input, it halts and outputs $f(n)$.

So, take f to be your favorite uncomputable function. Then there's a model of arithmetic such that in this model, the Turing machine computes f, at least when you feed the machine standard numbers as inputs.

So, very very roughly, there's a possible world in which your uncomputable function becomes computable!. *But you have to be very careful about how you interpret this result.*

The Trick

What's the trick? The proof is beautiful, but it would take real work to improve on Hamkins' blog article, so please read that. I'll just say that he makes extensive use of Rosser sentences, which say:

"For any proof of this sentence in theory T, there is a smaller proof of the negation of this sentence."

Rosser sentences are already mind-blowing, but Hamkins uses an infinite sequence of such sentences and their negations, chosen in a way that depends on the function f, to cleverly craft a model of arithmetic in which the Turing machine T computes this function on standard inputs.

But what's really going on? How can using a nonstandard model make an uncomputable function become computable for standard natural numbers? Shouldn't non-standard models agree with the standard one on this issue? After all, the only difference is that they have extra nonstandard numbers tacked on after all the standard ones! How can that make a Turing machine succeed in computing f on standard natural numbers?

I'm not 100% sure, but I think I know the answer. I hope some logicians will correct me if I'm wrong.

Read the Following Result Carefully

There is a Turing machine T with the following property. For any function f from the natural numbers to the natural numbers, there is a model of PA such that in this model, if we give T any standard natural n as input, it halts and computes $f(n)$.

When we say the Turing machine halts, we mean it halts after N steps for some natural number N. But this may not be a standard natural number! It's a natural number in the model we're talking about.

So, the Turing machine halts... but perhaps only after a nonstandard number of steps.

In short: you can compute the uncomputable, but only if you're willing to wait long enough. You may need to wait a nonstandard amount of time.

It's like that old Navy saying:

"The difficult we do immediately the impossible takes a little longer"

But the trick becomes more evident if you notice that one single Turing machine T computes different functions from the natural numbers to the natural numbers… in different models. That's even weirder than computing an uncomputable function.

The only way to build a machine that computes $n + 1$ in one model and $n + 2$ in another to build a machine that doesn't halt in a standard amount of time in either model. It only halts after a nonstandard amount of time. In one model, it halts and outputs $n + 1$. In another, it halts and outputs $n + 2$.

A scary possibility To dig a bit deeper and this is where it gets a bit scary we have to admit that the standard model is a somewhat elusive thing. I certainly didn't define it when I said this:

For you, the natural numbers are just $0, 1, 2, 3, ...$ with the usual way of adding and multiplying them. This is usually called the standard model of PA. The numbers 0, 1, 2, 3, … are called the 'standard' natural numbers.

The point is that "$0, 1, 2, 3, ...$" here is vague. It makes sense if you already know what the standard natural numbers are. But if you don't already know, those three dots aren't going to tell you!

You might say the standard natural numbers are those of the form $1 + ... + 1$, where we add 1 to itself some finite number of times. But what does 'finite number' mean here? It means a standard natural number! So this is circular.

So, conceivably, the concept of 'standard' natural number, and the concept of 'standard' model of PA, are more subjective than most mathematicians think. Perhaps some of my 'standard' natural numbers are nonstandard for you!

I think most mathematicians would reject this possibility... but not all. Edward Nelson tackled it head-on in his marvelous book Internal Set Theory. He writes:

Perhaps it is fair to say that "finite" does not mean what we have always thought it to mean. What have we always thought it to mean? I used to think that I knew what I had always thought it to mean, but I no longer think so.

If we go down this road, Hamkins' result takes on a different significance. It says that any subjectivity in the notion of 'natural number' may also infect what it means for a Turing machine to halt, and what function a Turing machine computes when it does halt.