What does it mean to say that no rational number squared is equal to 2 ? We might also speak of their being no integer solutions to $m^2 = 2n^2$, where $n \neq 0$.

Currently I think of math as something like the informal science of formal systems. Here I am talking about *real-world belief.* So I am not talking about formal proofs themselves or the symbols used in formal logic.

I lean at the moment to understanding the claim 2 has no rational square root that a certain calculation will never occur. If someone tells me they have found integers m and $n \neq 0$ such that $m^2 = 2n^2$, then I will find it impossible to believe them. I will instead believe that some mistake was made.

Define $f(n)$ as the number of 2s in n. So $f(20) = f(2^2 5^1) = 2$. Also $f(3) = 0$. And so on. This function is intuitively computable and it's easy to write a program. I emphasize intuitively computable because this intuition of computability is what affects our "real world" beliefs about real world calculations.

Note that $f(n^2) = 2f(n)$, because, for example, $f(20^2) = f(2^4 5^2) = 4 = 2f(20)$.

Likewise $f(2n) = 1 + f(n)$.

So $m^2 = 2n^2 \implies f(m^2) = f(2n^2) \implies 2f(m) = 1 + 2f(n)$. Here I use the $\implies$ for "psychological" implication. As in I find this reason "informally" convincing. If I imagine a nontrivial solution (excluding $m = n = 0$), then I can continue to imagine applying this computable function to both sides so that an even number "equals" an odd.

A skeptic might say "OK, so it turns out that even numbers can sometimes also be odd." I vaguely assume that some formal system could be created where formally even numbers could indeed sometimes be formally odd. But here I'm interested in "actual" belief about calculations.

I cannot "seriously" believe that an odd number can simultaneously be an even number.

Intuitionists allow for a proof of negation which is *not* a proof by contradiction. I learned the usual classical logic in school, so frankly it was all just proof by contradiction to me. Though I roughly understand the difference.

More generally, I am reluctant to separate proof from "truth." Yet here I talk about "real world" belief. This is why I am experimenting with understanding math as the necessarily *informal science* of formal systems.

One has to understand formal systems in order to study them informally. So mathematics is bigger than formal systems but also extremely concerned with them.

At the moment, I understand formal systems in terms of computable functions. But textbooks on computability use indirect proofs for fundamental results. This is why I find myself returning to a simple and famous indirect proof.

Let's assume that we have computable (total) function f such that $f(m, x) = 1$ if partial function $f_m(x)$ halts and $f(m, x) = 0$ if $f - m(x)$ does not halt. This is already a weird thing to do, because it's like assuming a nontrivial integer solution to $m^2 = 2n^2$. But we continue. Now I construct a partial function $g(x)$, using f as a subroutine, such that $g(x)$ halts iff $f_x(x)$ doesn't halt. It's a bit like getting this imaginary circuit board for f and adding a few pieces, in an intuitively easy way, to create g.

Keep in mind that we can enumerate the "code" for every partial recursive function. Basically we are just talking about all possible programs, each of which has a number computably determined by its

code. So $g(x)$ will halt if and only if $f_x(x)$ goes on forever.

Now we let k be the number of g, so $g = f_k$. This part is intuitively clear. Any program has a number, actually many numbers. But we can take the least such number. Now we consider $g(k) = f_k(k)$, and this computation halts only if $f_k(k)$ goes on forever. So we have a program that terminates and also goes on forever. Which is "nonsense." *Why* is it nonsense ? Because I personally can't make sense of a program that stops and never stops at the same "time" or during the same computation.

Intuitively, we have made a case that no single f can determine whether *any* given machine will stop. I think it's easy to make a case that there are programs that can detect easy-to-find looping behavior. So what is ruled out here is a one-size-fits-all divergence detector.

What real world change in my behavior might result from this "proof" ? I wouldn't waste my time, presumably, on searching for a *single* divergence detecting program. I can "see" that it is "impossible" in that it would "force" me to accept the idea of a non-terminating terminating program. It's my own insistence in avoiding nonsense beliefs that forces me to choose.

I might initially think that a program f is possible. But then I see this to-me-informal proof and decide that I can't make sense after all of the existence of such a program.

It's a big like a sci-fi, where a man travels back in time and murders his grandfather in the crib. Though here, at least, one can be "open-minded" about physics and time. It's harder perhaps to be so open-minded about calculations.

Someone claims to have f. So we build g. And then what ? If $f_k(k)$ halts, then something is wrong. If it doesn't halt, then... But what do we mean "if it doesn't halt" ?

The predicate Halts is *not computable.* That's the point. So Halts is not like Even or Odd. It's a vague, informal notion. What we have is Halted and NotHaltedYet. GoesOnForever is computable in some cases, at least if we agree that "obvious" loops can be built-in to the GoesOnForever detector, which still *cannot* work on *all* inputs.

So this slippery indirect proof is more ferocious than the irrationality of *sqrt*2, because to me it destroys classic logic. Or helps in the destruction.

I have indeed accepted informal beliefs that I might call "true." But my more general philosophical position is that "truth" has no deep meaning, that it is only a noise/mark for expressing situated (personal) belief.

I can believe that a string is a formal proof in a particular system. This belief is itself informal. Outside of the little formal system and part of the beliefs that I "live in."