

Re: Computability and logic

Source: <http://sci.tech-archive.net/Archive/sci.logic/2006-10/msg00208.html>

- *From:* "Tom" <tkorna@xxxxx>
 - *Date:* 1 Oct 2006 09:27:58 -0700
-

george wrote:

George, I apologize to you for all errors and unsubstantiated statements I ever make. I am very sorry, I am only trying to do some work, and I do err.

And thank you very much indeed for your kind attention and encouragement.

Tom wrote:

I was just thinking I am really trying to establish what Goedel's Bew(x,y) really is, you know.

It's just a defined binary first-order predicate.

Yes, indeed. Thank you again. (Why did I think it was _the second-order truth-function which finally turns out to have no finite formulation?)

In Godel's original it was in the language of PM but for our purposes it would be in the first-order language of Peano Arithmetic. If you use FOL with equality and 1 instead of s(.) then this is a VERY simple language;

Yes, _thank you very much indeed_, but I am on it at this particular moment. I am only trying to describe TM moves in FOPL. I see the relations between other systems quite clearly (although I will love working on the details a little later). Now it's the FOPL-TM relation and I need to have a really good think on it before I am competent enough to make use of your kind and most encouraging commentary.

its only terms are 0 and 1 and what you can build from them with + and x .
It doesn't have or need any predicates at all (since = is already built-in).

Re: Computability and logic

Yes. Still, afaik, that is the way around and you do run into the problems I mention just when you exclude = as a primitive and try to give some formulation of it.

But because that makes most statements in it every bit as unreadable as machine language (compiled into 0's and 1's), there is a mechanism for adding DEFINED predicates and functions. E.g. $\text{less}(x,y) \stackrel{\text{df}}{=} \exists z[(x+1)+y=z]$, or $\text{prime}(x) \stackrel{\text{df}}{=} \exists yz[\text{less}(y,2) \vee \text{less}(z,2) \vee \text{less}(x,y) \vee \text{less}(x,z) \vee \sim(y*z=x)]$

Yes, right.

Just as prime is defined in terms of less, here Bew is defined in terms of a high tower of other things that were defined earlier. But it is a NUMERIC function.

Right. So why did I think it was the truth–predicate which runs into infinite regress? I don't know. And, OK, anything is NUMERIC. That's so lovely, isn't it?

Strings are NOT actually involved here except to the extent that $(1 + (1 + (1 + (1 + 0))))$ is a string (which it isn't, actually; it's a term).

I see. Wouldn't you say that the term 'term' is an interpretation of the strings of the formal language underlying arithmetic?

I mean, I am trying to reach a single string which incorporates both the starting sequences and the rules of inference (Bew) from the inside,

You don't have to "reach" Bew; it was already defined in Godel.

OK. Let's be clear. Bew = provable, i.e. a theorem is an enunciation of a collection of theorems with respect to some inference rules(?). Isn't that: tape_B is the result of applying state_1 to tape_A ?

which is impossible.

Re: Computability and logic

Given that it's been done, it can hardly be impossible, unless you are saying that it was done in some non-string way, and what you are trying to do is translate it into a string.

My mistake, I am sorry.

But again, the framework is PA, is NUMBERS, NOT strings.

Yes. Still, I somehow adopt the formal approach (as described in e.g. Shapiro "Thinking about mathematics"). So I make a working assumption of numbers being interpretations of the formulas of the formal language underlying arithmetic.

Why would you be trying to redo FOL where the whole framework is strings and not terms? Aren't terms plenty string-like enough already?

I just need to do it all properly and see for myself.

If you can finesse the difference between a character and a string-of-length-1, you can already articulate some simple axioms about how strings work.

Yes, thank you for your patience, George.

But, again, if it is impossible how we do that at all?

I'm losing your antecedents of "it" and "that" here.

The theme is: how do I apply a state to a tape and return another state/tape? I am sorry.

I mean, given a TM description and a tape, how do you apply the description to the tape to obtain another tape. Don't you need a meta, and meta-meta, and meta-meta-meta ... machine to do that?

Well, no, me, personally, I don't. I can just look at it.

Yes. :-)

Re: Computability and logic

You do.

No, I don't. The fact that there is a TM that can confirm for me that the string II concatenated with the string III yields the string IIIII does NOT mean that I *need* this TM *before* I can do that.

So when we have a TM, we don't need to have it at all?

My brain is Turing-equivalent modulo finitude (despite the fact that Turing was a lot smarter than I am).

So is mine ever since I read the Turing original (despite the fact that you are a lot smarter than I am).

More to the point, mathematical functions in general don't "need" a TM-implementation *before* they can operate;

IMHO, implementation is description. We need _a description. Without description there is only intuition, and where intuition starts mathematics ends. So, with your kind permission, George, we need a description before anything happens.

they just operate on their arguments and produce the results they produce from them.

Instantaneously. For all arguments. IN PARALLEL. IN AETERNAM.

They're there, /I agree/.

Still, we manage to do the thing. How?

I didn't realize how old your question was. In the popular literature, this question occurs in Hofstadter's "Godel, Escher, Bach" in the context of how we determine that inference rules are sound.

I studied the whole book carefully. It merely scratched the surface as far as TM-FOPL relation is concerned so I will re-read its exposition

Re: Computability and logic

of soundness/completeness when I get more information. Actually, the problem I have at the moment is posed in the very first dialogue. It is solved further down the book by saying that the nature runs by itself. As I said, I am going to need a little more information (but, undoubtedly, I have ferreted out all that is there).

If we know P is non-contradictory, if we know $P \rightarrow Q$ is true, and we know modus ponens is sound, then, yes, we can infer Q , but how do we know that all the machinery we used in making THAT inference was sound?

Soundness proofs of infinite systems exceed my ability. I am sorry.

We can prove by truth-tables that if we have P , and we have $P \rightarrow Q$, then we can infer Q . But how do we know that the truth-table method is sound? Don't we need some further justification for that?

Yes, I see.

Hofstadter is continuing a thread begun by Lewis Carroll in "What the Tortoise Said to Achilles" some 80 years prior.

If you have children then one of them may at some point around age 4 have gotten into the habit, once you gave an explanation in response to a question of type "Why?", of iterating the question: "Why?"

There is in principle no end of that.

This is a lovely metaphor. I want to be able to tell them the truth about how to do something now that can (only) be done later. But I need to know that myself in the first place.

That is what AXIOMS are for.

Yes, and as was shown, because of the fact that we need to apply rules to them, whatever system that contains arithmetic runs into the difficulties described.

At some point you can have to be able to say that you can tell ad oculos, by inspection, that something

Re: Computability and logic

matches a pattern that makes it acceptable by fiat,

I realize that. Does the literature describe this moment?

just because we said so, just because we agreed NOT to seek DEEPER justification for things of THAT shape.

This is close. Thank you, George.

This raises 2 concerns. Either 1) we might be mistaken in recognizing the thing as matching the pattern (this is why axiom-sets have to be recursive, and why proof-predicates have to be PRIMITIVE-recursive), or 2) the reasoning might actually BE unsound and might lead us to accept something contradictory. In the case of first-order logic we rebut this objection by simply pointing out that since it is complete (the LOGIC, NOT a rich theory IN it) and since first-order consequence is r.e., then if we have accepted something that entails a contradiction, WE CAN PROVE that eventually, and thus evolve away from the contradiction.

Well, I think I will only be able to benefit from these great comments after I've studied Boolos&Jeffrey. I know they are truthful and valuable comments. Thank you very much again.

The problem in any case is NOT about "producing" strings or tapes.

Yes, I see. It reminds me of Chapter 2 (The Spirit of Truth) of Hodges's "Alan Turing".

TMs are thought of as just "having" their programs inside them somewhere

I gradually fail to be able to distinguish between a set of FOPL theorems and a TM. Actually, a tape is a NUMERIC entity, and so is the program, the result being a single NUMERIC entity (both explicit, if I have not omitted anything, which I may have). We end up trying to apply a single NUMERIC entity to itself, which is why we run into the infinite regress.

as you might know the recipe for baking a cake.

The fact that the recipe can also be written down in a cookbook

Re: Computability and logic

does NOT imply that you have to know how to read before you can know how to bake a cake.

Oh, please. _A description method/language is required before I start. This need not necessarily be natural language.

Thank you for posting again, George. I could do _no work without these conversations.

Kindest regards,
Tom

.