

Re: Computability and logic

Source: <http://sci.tech-archive.net/Archive/sci.logic/2006-09/msg00973.html>

- *From:* "Tom" <tkorna@xxxxx>
 - *Date:* 27 Sep 2006 11:55:55 -0700
-

MoeBlee wrote:

One method is to use "dummy variables". For example, suppose we want to replace the term t in the formula P with the term t' (P is a string and t and t' are strings). Then we choose (with certain technical restrictions) a dummy variable, say v (which is a string of exactly one symbol). Meanwhile, by recursion,

I am sorry, Moe. I tried to think about it as carefully /as I can/. It seems that at this point we assume comparison and substitution (i.e. substring replacement) to be predefined.

I can understand that it might seem that way, but if we were to start at the beginning, you'd see that by using recursion we avoid presupposition.

Moe, I am afraid I am not competent enough to pursue a more technical argument, much that I would indeed like. I take your word for my being mistaken about the whole thing, and very happily. Yet, the idea in my head remains which is that we are really speaking of the same thing in a different way. Unfortunately, I have no precise proof for this. Your position is that recursion is the key, I agree. I mean, anyone with at least a little sense does.

I'm not expert at this, but I have worked through some details. If I understand you correctly, your question is roughly the same question I was asking myself about a year ago, except my question was not general regarding any substrings whatever but rather substrings that are terms in formulas or that are subformulas in formulas.

:-)

This is the problem I asked myself about:

I understand how to recursively define substituting a term for a variable in a formula; and I understand how to recursively define substituting a formula for a sentence letter in a formula; but I don't know how to define substituting a term for a term in a formula nor substituting a formula for a formula in a formula.

Substituting a term for a variable or a formula for a sentence letter is a "many for one" substitution. We substitute a string that may be more than one symbol for just a single symbol. That's easy to define by recursion. But substituting a term for a term or a formula for a formula is a "many for many" substitution, and I couldn't figure out how to rigorously define that.

Then I thought that the best approach might not be so direct. Instead of directly defining substituting a term for a term, what I did instead was to COMPARE two different results of substituting a term for a variable. Suppose I want to substitute the term t' for the term t in the formula P . So I take the formula Q , where Q is just like P except Q is "guttled out" so that some variable v occurs where t occurred in P . Now, that might be where you think substitution is presumed as we are substituting v for t . But we don't actually substitute v for t . Rather, we reach even further back as if to START with the formula Q so that P is $Q[t|v]$. I know that doesn't seem right, but if I gave you the full technical context, you'd see it actually works out correctly without presupposition. So I substitute the term t for the variable v in the formula Q (notated as $Q[t|v]$) and that is actually just the formula P . And I substitute the term t' for the variable v in the formula Q (notated as $Q[t'|v]$). So (with certain other technical restrictions) $Q[t'|v]$ is the result of substituting t' for t in P . Then, by some more technical details (such as specifying that v is the first variable that does not occur free in P while Q is the unique formula such that P is $Q[t|v]$, blah blah blah), we can make this a formal definition. And the same method works, mutatis mutandis, for substituting a formula for a formula in a formula.

Notice that this provides for substituting t' for ALL occurrences of t in P . Sometimes we want to allow for substituting t' for only SOME occurrences t in P (for example, in stating a theorem schema of identity theory, we would want to say that the formulas $t=t'$ and P together imply any formula that is the result of t' substituted for t in zero or more (not necessarily all) places in P). But that too can be formalized with yet more technical rigmarole. However, to specify substituting for a PARTICULAR occurrence of t requires even more technical rigmarole.

Re: Computability and logic

Again, I will save and study it very carefully.

Another method is more general but more complicated (I haven't gone through the details myself). Say we want to substitute a string t' for a substring t in a string s (t , t' , and s being any strings, not necessarily terms and formulas). Then we note the positions that t occupies in s (e.g., in the string 'xyzuvwyzux', the substring 'yzu' occupies positions 3–5 and positions 8–10).

Here again, we pre-used substring replacement for computing both pairs of numbers. I.e. to determine the position of the target string we need to execute a number of comparisons. Those have to communicate by passing their results to subsequent stages, so substitution is involved, again, IMHO.

I don't follow you here.

It must be because of the terrible sloppiness of my thinking. I am sorry.

We're given a string of symbols. A string may be regarded as a function on a natural number (or often we "bump up" so that the domain excludes 0 so that we start at 1 instead of 0). So every entry in the string is indexed by a number, which is the position of that entry in the string. So I can say that in the string 'xyzuvwyzux', we have the substring 'yzu' occurring in positions 3–5 and 8–10 (except that, technically, the domain of the string 'yzu' is not $\{1\ 2\ 3\}$ but rather $\{3\ 4\ 5\}$ and $\{8\ 9\ 10\}$, and we can "adjust" those back down to $\{1\ 2\ 3\}$). There are no presuppositions about substitution so far. Then we pull the whole thing apart, using POSITIONS to keep track of where substrings start and end, then put it all back together (concatenation, which can be rigorously defined) with different values for some of the positions. Still, no presupposition about substitution.

Well, you added a little more description here. I need time to study it.

If you don't mind me asking at this point, what in your opinion presupposes comparison?

Re: Computability and logic

By the way, you'll find a lot of the formalization regarding syntax for first order languages is done in the proof of incompleteness.

Thank you. I will surely use that information.

That's where you usually find the explicit recursive functions that are only informally given in earlier chapters of a logic textbook. You don't usually find a formalization of the GENERAL function of replacing a string for a string in a string, but at least you'll find the recursive definition of $P[t|v]$, which is the most important instance as far as syntax for first order languages is concerned.

Yes, I see.

By all means. You number-theorize your constructors and selectors. Then you reduce arithmetic to FOPL. Then you goedelize logic. The result being a single natural number is re-applied to itself.

Okay, so I see you already knew about what I just mentioned.

Oh, I just gave you a large-scale map in my mind. I need to do some more hiking to fill it in.

Most people don't pursue questions of rigor to the extent that you are doing.

IMHO, it's because they usually do it with some implementation in mind, and as soon as the description is good enough their job is done.

Yes, for example, if one had the assignment of writing a computer program to verify all these syntactical operations, then one would have to come to grips with such technicalities as rigorous definitions for substitutions.

"The Spirit of Truth" is my favourite chapter of my most favourite book ever: A. Hodges, "Alan Turing".

I think you'll find that if you hold to that demand, then there are many steps that

Re: Computability and logic

you have to fill in for yourself that the textbooks
(understandably)
take for granted as formal steps we could complete though
tedious to do
so.

But I love doing that! After all, nothing else is there, or is there?

Well, that's a subject for some philosophical debate!

Well, IMHO, the Church–Turing thesis /holds/ philosophy really tight.

But I think most people at least agree that it helps to understand not just every
technical step but also to have an intuitive understanding of what it
all adds up to.

Yes, indeed. And Moe, I can not tell you how grateful I am to you for
taking this while with me. Thank you very much for your patience and
attention.

Kindest regards,
Tom

.