

Re: Cantor For Dummies ...

Source: <http://sci.tech-archive.net/Archive/sci.math/2007-03/msg05841.html>

- *From:* briggs@xxxxxxxxxxxxxxxxxxxx
 - *Date:* 29 Mar 2007 13:04:09 -0500
-

In article <1175183785.561962.104910@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>, "georgie" <geo_cant@xxxxxxxx> writes:

On Mar 29, 9:55 am, riderofgiraffes <mathforum.org...@xxxxxxxxxxxxxxxx> wrote:

I never said that the list being fed in was the output list. Never once did I say that an output list was being used as input. I'd be interested to see where you claim I said that an output list was being used as input.

You said that your algorithm can use ANY input.

Yes. I said that if you provide a list of committees, then my algorithm can and will produce a committee that's not on your list.

No it can't. It can't use a list that contains your algorithm's output.

Why not?

If you start with a list L1 of committees and then apply my algorithm to that, we get a new committee C1 that wasn't on the original list.

Now create a list L2 which is C1 prepended to L1.

Tell me why we can't now apply my algorithm to L2.

Why would that change the fact that each L_n is the output of an algorithm? Running some algorithm in a loop n times is still an algorithm and therefore

is contained in the collection of all algorithms that output a committee.

There is indeed a collection of all algorithms that output a committee. That much is true. And that does mean that for any algorithm you can generate and any algorithm Rider can answer back with, there will be an algorithm in that collection that produces a list containing his committee tacked onto your list.

You're absolutely correct in that intuition.

[It turns out that this collection is countable. Which is an immediate consequence of the convention that all "algorithms" are finitely describable]

However, it turns out that is no algorithm that lists all and only the algorithms in that collection. There are algorithms that list all the algorithms in the collection. And there are algorithms that list only algorithms in the collection. But none that do both.

[This is a well known result in computability theory and follows from a simple diagonalization argument. But since Cantor appears in the thread title, I don't expect that to carry any weight. Maybe we should rename this to "Turing for dummies"]

Having a collection of algorithms isn't good enough to generate a list of committees with. An actual algorithm is what we're after. Any such algorithm can either generate a complete list of committees or an error-free list of committees. It can't do both.

Rider's algorithm depends on the implicit assumption that your algorithm is error-free. If algorithm number 31,416 in your list of algorithms doesn't actually produce an answer when asked whether member 31,416 is in committee number 31,416 then Rider's algorithm can't decide whether member 31,416 is or is not in the committee that it produces.

On the assumption that your algorithm is error-free, Rider has already succeeded in delivering a proof that your algorithm is incomplete.

.