

Re: Alan Turing's Halting Problem is Incorrect (FINAL PART)

Source: <http://sci.tech-archive.net/Archive/sci.logic/2004-07/1274.html>

From: Acme Diagnostics (*LFinezapthis_at_partpostmark.net*)

Date: 07/11/04

Date: 11 Jul 2004 03:20:14 -0500

Chris Menzel <cmenzel@remove-this.tamu.edu> wrote:

>>

>> http://www.csee.umbc.edu/~squire/cs451_l26.html

>>

>> *Is diagonal.c a good demonstration of the Halting Problem on*

>> *real computers?*

>

>*I'd put it a bit differently: it's a good demonstration of the*

>*unsolvability of the Halting Problem for a real programming language.*

That's what I meant. Thanks. This question was important to me.

>*The Halting Problem isn't really about computers,*

In that case, I really have no interest in it. I have enough trouble with computers. <g>

>*at least not directly,*

>*but about the class of questions we are capable of answering*

>*algorithmically. Of course, this has *implications* for computers --*

>*since there is no algorithm for solving the Halting Problem, no*

>*computer, real or imagined, can be programmed to solve it.*

I have no interest in solving it, believe me!

>> *Is there any *linkable* proof of the Halting Problem for *real*

>> *computers* that does not contrive a logic paradox to make the proof?*

>

>*I'm afraid I don't really understand the question. First, as noted, the*

>*emphasis on "real computers" here is a red herring. Second, there is*

>*nothing contrived about the proof, and in particular there is no logical*

>*paradox. A paradox is an argument in which a contradiction follows from*

>*premises that we find intuitively true. Genuine paradoxes are*

>*disturbing, as they show that one has a false belief, as true premises*

>*cannot entail a contradiction.*

Ok. That's why I used the word "paradox." If there is no contrived paradox, then that would answer my question well enough. No link is necessary as diagonal.c becomes the link to a program with no contrived paradox.

>The proof of the unsolvability of the
>Halting Problem also involves an argument to a contradiction -- it is,
>in fact, an instance of the general *method* of proof by contradiction,
>a.k.a. "reductio ad absurdum", a ubiquitous form of reasoning used
>throughout the history of mathematics: Infer not-A, if the assumption
>that A is true leads to a logical absurdity. Thus, for instance, in the
>usual proof that $\sqrt{2}$ is irrational, we assume that it is not, i.e.,
>that there are relatively prime integers m, n such that $\sqrt{2} = m/n$.
>We then deduce a contradiction (e.g., that m and n are not relatively
>prime after all), and conclude that our assumption was false. And in
>the case of the Halting Problem: Assume you've got a program H that can
>tell you if an arbitrary program P halts on input I . On that
>assumption, it follows that you can construct a program (*Diagonal.c*)
>that halts on a certain input (viz., *Diagonal.c* itself) iff H says it
>doesn't halt on that input. So our assumption that there is such a
>program as H must be false. The difference between this and a genuine
>paradox, then, is that the assumption from which the contradiction
>follows does not have any particular intuitive hold on us ahead of time;
>whether there is such a program as H just seems like an open question,
>which the proof then answers. Hence, the argument in the proof is no
>genuine paradox.

Sorry to prompt so much explanation. On a trivial intuitive level, it seems to me that a logical impossibility is intentionally created. In other words, on cursory inspection I think most would wonder if a compiler could magically check itself for infinite loops if only the logic in diagonal.c was reversed. But I'm way too experienced with programming to have faith that something "intuitively obvious" will work when self-reference is involved, not to mention an imaginary, e.g. "Halt()" function, at least without spending a great deal of time on it. I'm fairly sure that a real "Halt()" function will never be written in any useful time-frame, and I'm quite sure I will never run into this Halting Problem.

Larry