

Re: Can returning a value change the value itself (in the Halting Problem)

Source: <http://sci.tech-archive.net/Archive/sci.logic/2004-08/3035.html>

stephen_at_nomail.com

Date: 08/24/04

Date: 24 Aug 2004 17:34:32 GMT

In comp.theory Marc Goodman <marc.goodman@comcast.net> wrote:

: Martin Shobe wrote:

:> Of course the value changed. The *input* changed.

: The context is a TM with access to its own state transition
: table that can calculate the check sum of its own state
: transition table. Do you finally see that even making
: a "trivial" change to such a TM can cause the result produced
: by such a TM to change? Or do we have to go through this
: line of argument three more times?

: But, both

:> functions implement exactly the same, um, function.

: You are free to call it "different input" if you want,
: but the fact remains that if you are talking about
: a TM with access to its own state transition table, the
: line between "input" and "process" has been blurred.
: To the extent that a seemingly local change to the
: TM can have a global change to the result produced by
: the TM.

The point of a computational model is to compute mathematical functions. A mathematical function has an input and an output. There is no blurring, other than in your presentation. For your model where the program can inspect itself you have to decide if the program is part of the input or not. If the program is not part of the input, then in checksum example your program computes a constant function, in which case it is easy to write another program that computes the same function. If the program is part of the input, then it is also easy to write another program that computes the same function. So what mathematical function are you trying to compute?

The whole reflective code idea really does not affect the Halting problem at all. Consider a reflective version of Halt(x,y). This program can inspect its own code. However, its own code

sci.logic: Re: Can returning a value change the value itself (in the Halting Problem)

is a constant, so any conditionals based on its own code are independent of the input. I could therefore write a function `Halt2(x,y)` that replaces all the reflective conditionals with a hard coded true or false. I then create `LoopIfHalts2(x)`, and `Halt(LoopIfHalts2,LoopIfHalts2)` is guaranteed to return the wrong answer.

Stephen