

Re: Rado's Sigma and the Halting Problem for Programs

Source: <http://coding.derkeiler.com/Archive/General/comp.theory/2004-08/2061.html>

From: peter_douglass (*baisly_at_gis.net*)

Date: 08/11/04

Date: 11 Aug 2004 13:35:30 -0700

"Acme Diagnostics" <LFinezapthis@partpostmark.net
<mailto:LFinezapthis@partpostmark.net>> wrote ...

AD > Is there any *linkable* proof of the Halting Problem for a
AD > computer language that does not contrive a logic paradox
AD > and require self-reference to make the proof?
AD > I'm not interested in Turing Machines at all, but only the
AD > proof "for a computer language."

to which

"r.e.s." <r.s@ZZmindspring.com <mailto:r.s@ZZmindspring.com>> in
message <news:5VxRc.13662\$K.1818@newsread2.news.pas.earthlink.net>...
responded:

RES > I doubt there's a proof that avoids reductio ad absurdum...

I'm not sure what a *linkable* proof is. However, I believe the
Halting Theorem may be proven without recourse to reductio
ad absurdum. Here is my stab at it.

Assume as given the following:

$G :: N \rightarrow TM, G' :: TM \rightarrow N$ (an isomorphism between natural numbers
and Turing Machines)

$\langle _ , _ \rangle :: N \times N \rightarrow N$ (a bijection between pairs of naturals and
naturals)

$Eval :: TM \times N \rightarrow N + _ _$

(an evaluation function such that $Eval(M,x)$ is the result of running
machine M on the input tape consisting of a representation of x .

We will assume that the result is converted to a natural number if the
computation halts, and is bottom, i.e. $_ _$ if the computation diverges
)

(Construction)

Assume that from a Turing Machine M we can construct a Turing Machine M' such that the following holds:

```
Eval(M',x) == begin
    if Eval(M, <G'(M'),x>) == "no"
        then return "yes"
        else loop_forever() ;
    end
```

Definitions:

(Totality) Machine M is total

iff forall x , $Eval(M,x) \neq _ _$

(Partiality) Machine M is partial

iff There exists an x such that $Eval(M,x) = _ _$

(Halts)

$Halts :: TM \times N \rightarrow Bool$. That is,

$Halts(M,x) = True$ if the Turing Machine M halts when given an input tape containing a representation of x

$Halts(M,x) = False$ otherwise.

Note that $Halts(M,x)$ is a well defined function.

It is not the hypothetical Turing machine "Halts" that is proven not to exist by reductio ad absurdum in most popular proofs of the Halting Theorem.

(Implements)

Machine M partially implements Halts

iff $Eval(M, \langle x,y \rangle) \neq _ _ \Leftrightarrow Eval(M, \langle x,y \rangle) = Halts(G(x), y)$

(That is, if M halts on input $\langle x,y \rangle$ then the value returned is

$Halts(G(x),y)$)

CLAIM:

Our claim is that for any G and for any Turing Machine M either M does not partially compute Halts, or M is not total.

Thus, there is no Turing Machine which totally computes Halts. (This is a restatement of the Halting Theorem)

PROOF:

When M' (defined in Construction) is given $G'(M')$ as input it either halts or it diverges.

Case 1. $Eval(M',G'(M')) \neq _ _$ (i.e. halts)

(Construction) $\Rightarrow Eval(M, \langle G'(M'), G'(M') \rangle) = "no"$

(Implements) $\Rightarrow M$ does not partially implement Halts

Case 2. $Eval(M',G'(M')) = _ _$ (i.e. diverges)

(Construction) \Rightarrow either $Eval(M, \langle G'(M'), G'(M') \rangle)$ diverges

or $Eval(M, \langle G'(M'), G'(M') \rangle)$ returns a value $\neq "no"$

comp.theory: Re: Rado's Sigma and the Halting Problem for Programs

Case 2.a. $\text{Eval}(M', G'(M')) = _ _ _ \wedge \text{Eval}(M, \langle G'(M'), G'(M') \rangle) = _ _ _$
(Totality) $\Rightarrow M$ is not total

Case 2.b. $\text{Eval}(M', G'(M')) = _ _ _$
 $\wedge \text{Eval}(M, \langle G'(M'), G'(M') \rangle)$ returns value \neq "no"
(Implements) $\Rightarrow M$ does not partially implement Halts

QED