

Re: The proof that I was referring to is on the website

Source: <http://sci.tech-archive.net/Archive/sci.logic/2004-08/1773.html>

From: Martin Shobe (*mshobe_at_sbcglobal.net*)

Date: 08/13/04

Date: Fri, 13 Aug 2004 04:24:20 GMT

On Thu, 12 Aug 2004 12:09:50 GMT, "Peter Olcott"
<olcott@worldnet.att.net> wrote:

>
> *"Martin Shobe" <mshobe@sbcglobal.net> wrote in message
news:020kh0dit2umrn2tk6lih0q91mnqh0u38u@4ax.com...*
>> *On Wed, 11 Aug 2004 01:59:57 GMT, "Peter Olcott"*
>> *<olcott@worldnet.att.net> wrote:*
>>
>> >
>> > *"David C. Ullrich" <ullrich@math.okstate.edu> wrote in message
news:h77hh09dgpr36g7q1g671b2m9sphqkehfr@4ax.com...*
>> >> *On Mon, 09 Aug 2004 23:43:29 GMT, "Peter Olcott"*
>> >> *<olcott@worldnet.att.net> wrote:*
>> >>
>> >> >
>> >> > *"Simon G Best" <s.g.best@btopenworld.com> wrote in message
news:411773FB.3020801@btopenworld.com...*
>> >> >> *Peter Olcott wrote:*
>> >> >> >
>> >> >> > *If Turing use the diagonalization approach then the basis of my refutation
>> >> >> > would not apply because the pure math version does not permit the
>> >> >> > equations to have any intelligence. The equations are not allowed to
>> >> >> > refrain from returning a result.*
>> >> >>
>> >> >> *It was all about maths to begin with. Did you really not know that?*
>> >> >>
>> >> >> *Simon*
>> >> >>
>> >> > *No I did not. I thought that it was about Turing Machines.*
>> >>
>> >> *huh? tms are not a mathematical topic?*
>> >>
>> > *Not really they are much more of a comp.theory topic.*
>>
>> *I believe that nowadays, they aren't that much of a computer theory*

sci.logic: Re: The proof that I was referring to is on the website

>> *topic. The preferred formalism in computer theory is currently the
>> lambda calculii.*
>>
>> *Martin*
>>
> *Yet I have shown that the purely mathematical approach sometimes
> ignore crucial details that could have be applied to solving a problem.*
> *Specifically because these details are ignored, a solution is PRESUMED
> to be impossible.*
>

The purely mathematical approach ignores the things you have come up with because they really are irrelevant. For instance, let's take your idea of a caller ID.

You construct a "halt analyzer"...

```
#define HALTS 1
#define DOESNT_HALT 0
#define THBPT -1

int WillHaltPO(char const * machine, char const * input)
{
    bool safeToReturn;
    int result;

    /* code block A */
    safeToReturn = QueryContext(...);
    /* code block B */

    return result;
}
```

Now we will grant your assumption that QueryContext(...) returns true whenever it is "safe" to return the results of WillHaltPO and 0 otherwise.

Now we will construct a second function.

```
int Naughty(char const * machine, char const * input)
{
    bool safeToReturn;
    int result;

    /* code block A */
    safeToReturn = true;
    /* code block B */

    return results;
}
```

Re: The proof that I was referring to is on the website

And a third

```
int Bad(char const * machine)
{
  if (Naughty(machine, machine) == HALTS)
  {
    while (1);
  }

  return 0;
}
```

Now, we execute your machine in a safe context,
machine input safeToReturn result
upon entry Bad Bad ??
after code block A Bad Bad ??
after QueryContext Bad Bad true ?
after code block B Bad Bad true HALTS

However, we now see that the trace of Bad is as follows.
machine
upon entry Bad
before Naughty Bad

[Naughty] machine input safeToReturn result
upon entry Bad Bad ??
after code block A Bad Bad ??
after QueryContext Bad Bad true ?
after Code block B Bad Bad true HALTS

and now we see that Naughty doesn't halt.

Ok. Let's try the other answer.

```
machine input safeToReturn result
upon entry Bad Bad ??
after code block A Bad Bad ??
after QueryContext Bad Bad true ?
after code block B Bad Bad true DOESNT_HALT
```

However, we now see that the trace of Bad is as follows.
machine
upon entry Bad
before Naughty Bad

[Naughty] machine input safeToReturn result
upon entry Bad Bad ??
after code block A Bad Bad ??
after QueryContext Bad Bad true ?
after Code block B Bad Bad true DOESNT_HALT

sci.logic: Re: The proof that I was referring to is on the website

And now we see that Naughty halts.

Either way, WillHaltPO gets it wrong.

Martin