

Re: Lambda Calculus and Turing Equivalence

Source: <http://coding.derkeiler.com/Archive/General/comp.theory/2004-12/0114.html>

From: Stephen Harris (cyberguard1048-usenet_at_yahoo.com)

Date: 12/02/04

Date: Thu, 02 Dec 2004 16:54:05 GMT

<gds@best.cut.here.com> wrote in message
news:Bqyrd.28629\$NC6.6974@newsread1.mlpsca01.us.to.verio.net...
> At 1 Dec 2004 12:04:41 -0800, examachine@gmail.com (Eray Ozkural exa)
> wrote:
>> torbenm@diku.dk (Torben Ægidius Mogensen) wrote in message
>> news:<7zsm6qvyqk.fsf@pc-032.diku.dk>...
>>> A TM does not have an *_infinite_* tape, it has an *_unbounded tape_*.
>>> This means that the tape is guaranteed to be as large as you need it
>>> or, equivalently, that the tape will grow as you need it but always
>>> stay finite.
>>>
>>> It isn't hard to write a TM simulator in pure lambda calculus. The
>>> converse is harder, but possible.
>>
>> Precisely my point.
>>
>> I am saying to those who say that PCs are not TMs, like Chapman and
>> Harris, that they do not know the difference between unbounded
>> (potentially infinite) and actually infinite. (Although they claim
>> that they do) The latter is a set theoretical notion that is not
>> required to depict TMs, I doubt this distinction was so clear back
>> then.
>
> Part of the problem is that there is a fair amount of literature that
> describes TMs as having infinite memory, e.g. the second paragraph of
> http://en.wikipedia.org/wiki/Turing_machine. (Arguably, this is an
> informal description.)
>
> I haven't read all of the articles in this and related threads, but so
> far none that I have seen have commented on the difficulty TMs have in
> modeling "ongoing computation," such as what is found in operating
> systems. (The Wikipedia article mentioned above has a short
> discussion of this.)
>
> --gregbo
> gds at best dot com

This issue is strictly one of knowing the definition. Turing said the TM can compute Pi which is infinitely long, one finite digit at a time. All the physical memory in the universe, if every subatomic particle could be used, is not sufficient to accomplish the storage Turing's tape can do. It doesn't matter if you consider the tape to be only hugely finitely large. Turing's postulated tape provides more memory storage than any amount of memory available utilizing the entire physical universe. It is just a matter of definition, not philosophical. There are countless websites describing the TM, as ideal, as hypothetical, as theoretical, as abstract, IOW, not physical.

One can simulate a TM on a PC, or build a TM, except for the tape which always will have less potential capability than the tape that Turing described. The tape Turing employs is not real, nor can it be real.

http://en.wikipedia.org/wiki/Turing_machine

Note that every part of the machine is finite, but it is the potentially unlimited amount of tape that gives it an unbounded amount of storage space.

<http://plato.stanford.edu/entries/turing-machine/>

In modern terms, the tape serves as the memory of the machine, while the read-write head is the memory bus through which data is accessed (and updated) by the machine. There are two important things to notice about the definition. The first is that the machine's tape is infinite in length, corresponding to an assumption that the memory of the machine is infinite. The second is similar in nature, but not explicit in the definition of the machine, namely that a function will be Turing-computable if there exists a set of instructions that will result in the machine computing the function regardless of the amount of time it takes. One can think of this as assuming the availability of infinite time to complete the computation.

These two assumptions are intended to ensure that the definition of computation that results is not too narrow. This is, it ensures that no computable function will fail to be Turing-computable solely because there is insufficient time or memory to complete the computation. If a function is not Turing-computable it is because Turing machines lack the computational machinery to carry it out, not because of a lack of spatio-temporal resources.

These are called abacus computers by Lambek (Lambek 1961), and are known to be equivalent to Turing machines.

The modern digital computer is subject to finiteness constraints that we have abstracted away in the definition of abacus machines, just as we did in the case of Turing machines. Physical computers are limited in the number of memory locations that they have, and in the storage capacity of each of those locations, while abacus machines are not subject to those constraints. Thus some abacus-computable functions will not be computable by any physical machine. (We won't consider whether Turing machines and modern digital computers remain equivalent when both are given external inputs, since that would require us to change the definition of a Turing machine.)