> Date: Tue, 20 Feb 90 13:32:38 EST
> From: "William J. Rapaport" <rapaport@cs.buffalo.edu>
> To: case@cs.buffalo.edu
> Subject: a philosophical question
> Cc: rapaport@cs.buffalo.edu
> Resent-Date: Tue, 20 Feb 90 13:37:24 EST
> Resent-From: case@udel.edu
> Resent-To: case@cis.udel.edu
>
> John-
>
> Here's a phil[o]sophical/computational question the answer to which will
> determine how I respond to a paper I'm commenting on at an APA meeting.
> I'll pose the question in broadly general, perhaps even hopelessly
> vague, terms, rather than linking it to the particular topic of the paper,
> but I'll fill you in on details if you want/need them.
>
> Given a Turing machine program for computing, say, the greatest common
> divisor of 2 integers, and given 2 integers as input, the program will
> output the gcd.
>
> Given a universal TM and, as input, the TM for gcd and 2 integers, what
> does the UTM "do":
>
> 1. does it (merely) run a (virtual) TM (which, in turn, computes a gcd)?

That depends on which universal TM does it. Some may do that; others might just run mysterious code (MC) which happens to compute gcd, but the MC could be so obscure that one cannot prove in **ZF** that it does the job. Furthermore, one may not be able to prove in **ZF** that the machine is universal even though it is, ... My point is, in part, that universal machines are behaviorally correct, possibly, but not necessarily, "intensionally" correct (intensionally correct, in that they do 1). *N.B.* Of course a given user of the term 'universal TM' may or may not mean what I do, may or may not be capable of thinking clearly about it after reading this paragraph and seeing me construct an example which runs MC, may understand me but not want to mean what I mean even after they understand me, ... Most universal TMs in the literature will satisfy 1 simply because that is an obvious way to get one, and most TM literature will be on about the mere existence of universal TMs. The literature rarely gets beyond that. There is an old *IEEE Transactions on Computers* paper by Andy Kang[Kan75] which deals with somewhat non-standard examples. I have a paper [Cased] which references Kang and which also contains non-standard examples. More to the point regarding your queries, I'll indicate below a neat example due to me and generated by your queries. If it is helpful, you may use it with credit, in a joint paper, ...

```
> 2.  does _it_ compute the gcd?
```

You bet it does, but it may or may not do it by an obvious translation of Euclid's algorithm or whichever gcd algorithm it is given as input.

```
>
> -Bill
```

Neat Example From John Case

Let N denote the set of natural numbers, $\{0, 1, 2, \ldots\}$. Let $\langle \cdot, \cdot \rangle$ denote any fixed *pairing function* [Rog67, Roy87], i.e., a computable, bijective mapping: $N \times N \rightarrow N$. Suppose φ is a standard acceptable system (numbering) [Rog67, Rog58, MY78, Ric80, Ric81, Roy87], based on TMs, for computing all the partial computable (partial recursive) functions: $N \rightarrow N$. u is *universal* $\stackrel{\text{def}}{\Leftrightarrow} (\forall p, x)[\varphi_u(\langle p, x \rangle) = \varphi_p(x)]$.

Let W_p denote the domain of φ_p. Hence, $\{W_p \mid p \in N\}$ is the class of all r.e. sets.

Suppose e_0 is φ-program naturally expressing Euclid's gcd algorithm. Suppose Φ is any Blum Complexity Measure [Blu67, MY78] associated with φ. We say that, for partial functions η and θ, $\eta =^* \theta \stackrel{\text{def}}{\Leftrightarrow} \text{card}(\{x \mid \eta(x) \neq \theta(x)\}) < \infty$.

Now then, since the axoms of **ZF** form a recursive set of wffs, the set of theorems derivable in **ZF** form an r.e. set. For each meta-assertion of the form

$$[\varphi_u(\langle e_0, \cdot \rangle) = \varphi_{e_0}(\cdot)], \tag{1}$$

for $u \in N$, effectively pick a *natural* cwff, $\mathcal{F}_u$, of **ZF** expressing it. Clearly, then,

$$(\exists z_0)[W_{z_0} = \{u \mid \mathcal{F}_u \text{ is provable in } \mathbf{ZF}\}].$$

By implicit, informal application of the Kleene Recursion Theorem in φ [Rog67, Page 214] (see also [Roy87]), there is a φ-program, u_0, of course expressing a corresponding TM program, whose "top level" behavior we describe just below.

> **begin** $\approx u_0$
> On input $\langle p, \langle x, y \rangle \rangle$, u_0 first creates a quiescent copy of itself[1] to use as data, and, then, checks whether $[p = e_0 \ \wedge \ \Phi_{z_0}(u_0) \leq \langle x, y \rangle]$. If so, u_0 goes into an infinite loop (thereby *failing* to simulate the behavior of $p = e_0$ on input $\langle x, y \rangle$). If not, u_0 emulates p on $\langle x, y \rangle$ (thereby computing $\varphi_p(\langle x, y \rangle)$).
> **end**

Case(1). $\mathcal{F}_{u_0}$ is provable in **ZF**. Then $u_0 \in W_{z_0}$; hence,

$$(\exists \langle x_0, y_0 \rangle)(\forall \langle x, y \rangle \geq \langle x_0, y_0 \rangle)[\Phi_{z_0}(u_0) \leq \langle x, y \rangle].$$

[1] This is the part of u_0 that informally exploits the self-reference allowed by the Kleene Recursion Theorem.

It follows from this (and the behavior of u_0) that $\lambda\langle x,y\rangle.\varphi_{u_0}(\langle e_0,\langle x,y\rangle\rangle) =^* \lambda\langle x,y\rangle.\uparrow$. However, φ_{e_0} is total. Therefore, meta-assertion (1) above is *false* for $u = u_0$. Hence, $\mathcal{F}_{u_0}$ is a false assertion, provable in **ZF**. This is a contradiction[2].

Hence, we must have

Case(2). $\mathcal{F}_{u_0}$ is *not* provable in **ZF**. Then u_0, clearly, *is* universal for φ; hence, it correctly simulates e_0, yet one cannot prove in **ZF** that it properly simulates e_0.[3]

In case we need it: it should be clear that many variants of u_0 above can be constructed to prove corresponding variants of *Case (2).*

References

[Blu67] M. Blum. A machine–independent theory of the complexity of recursive functions. *Journal of the Association for Computing Machinery*, 14:322–336, 1967.

[Cased] J. Case. Operator speed-up for universal machines. *IEEE Transactions on Computers*, accepted.

[Kan75] A. Kang. On the efficiency of universal machines. *IEEE Transactions on Computers*, 1010–1012, 1975.

[Men79] E. Mendelson. *Introduction to Mathematical Logic.* Van Nostrand, 1979.

[MY78] M. Machtey and P. Young. *An Introduction to the General Theory of Algorithms.* North–Holland, 1978.

[Ric80] G. Riccardi. *The Independence of Control Structures in Abstract Programming Systems.* PhD thesis, State University of New York at Buffalo, 1980.

[Ric81] G. Riccardi. The independence of control structures in abstract programming systems. *Journal of Computer and System Science*, 22:107–143, 1981.

[Rog58] H. Rogers. Gödel numberings of the partial recursive functions. *Journal of Symbolic Logic*, 23:331–341, 1958.

[Rog67] H. Rogers. *Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967. Reprinted. MIT Press. 1987.

[Roy87] J. Royer. *A Connotational Theory of Program Structure. Lecture Notes in Computer Science, 273*, Springer–Verlag, 1987.

[2]Our construction applies, not just to **ZF**, but to any recursively axiomatized [Men79] theory in which one can express all the meta-assertions (1) above, *but cannot prove* any false ones.

[3]Of course, one can prove in a mere fragment of first-order arithmetic that any *standard, textbook*, universal φ-program really is universal, and, hence, in particular, that it simulates e_0.