

# Can Turing machines capture everything we can compute?<sup>1</sup>

Bhupinder Singh Anand<sup>2</sup>

If we define classical foundational concepts constructively, and introduce non-algorithmic effective methods into classical mathematics, then we can bridge the chasm between truth and provability, and define computational methods that are not Turing computable.

## Contents

1. Introduction
2. Non-constructivity in classical theory
3. Constructive definitions
4. Effective solvability of the Halting problem
5. Is there a case for an Arithmetical Provability Thesis?

## 1. Introduction

In a short, 2003, opinion paper, “Computation Beyond Turing Machines” [WG03], Peter Wegner and Dina Goldin advanced the thesis that:

“A paradigm shift is necessary in our notion of computational problem solving, so it can provide a complete model for the services of today's computing systems and software agents.”

---

<sup>1</sup> Created: Thursday 3<sup>rd</sup> June 2004 5:44:23 PM IST. Updated: Thursday 2<sup>nd</sup> November 2006 9:17:20 AM IST by re@alixcomsi.com. A .pdf file of this essay before the current update is available at <http://arxiv.org/abs/math/0304379>.

<sup>2</sup> The author is an independent scholar. E-mail: re@alixcomsi.com; anandb@vsnl.com. Postal address: 32, Agarwal House, D Road, Churchgate, Mumbai - 400 020, INDIA. Tel: +91 (22) 2281 3353. Fax: +91 (22) 2209 5091.

We note that Wegner and Goldin's arguments, in support of their thesis, seem to reflect an extraordinarily eclectic view of mathematics, combining both acceptance of, and frustration at, the standard interpretations and dogmas of classical mathematical theory:

(i) ...Turing machines are inappropriate as a universal foundation for computational problem solving, and ... computer science is a fundamentally non-mathematical discipline.

(ii) (*Turing's*) 1936 paper ... proved that mathematics could not be completely modeled by computers.

(iii) ...the Church-Turing Thesis ... equated logic, lambda calculus, Turing machines, and algorithmic computing as equivalent mechanisms of problem solving.

(iv) Turing implied in his 1936 paper that Turing machines ... could not provide a model for all forms of mathematics.

(v) ...Gödel had shown in 1931 that logic cannot model mathematics ... and Turing showed that neither logic nor algorithms can completely model computing and human thought.

These remarks vividly illustrate the dilemma with which not only Theoretical Computer Sciences are faced, but all applied sciences that depend on mathematics for providing a verifiable language to express their observations precisely:

Are formal classical theories essentially unable to adequately express the extent and range of human cognition, or does the problem lie in the way formal theories are classically interpreted at the moment? The former addresses the question of whether there are absolute limits on our capacity to express human cognition unambiguously; the latter, whether there are only temporal limits - not necessarily absolute - to the

capacity of classical interpretations to communicate unambiguously that which we intended to capture within our formal expression.

Prima facie, applied science continues, perforce, to interpret mathematical concepts Platonically, whilst waiting for mathematics to provide suitable, and hopefully reliable, answers as to how best it may faithfully express its observations verifiably.

This dilemma is also reflected in Fortnow's on-line rebuttal<sup>3</sup> of Wegner and Goldin's thesis, and of their reasoning. Thus Fortnow divides his faith between the standard interpretations of classical mathematics (and, possibly, the standard set-theoretical models of formal systems such as standard Peano Arithmetic), and the classical computational theory of Turing machines. He relies on the former to provide all the proofs that matter:

“Not every mathematical statement has a logical proof, but logic does capture everything we can prove in mathematics, which is really what matters”;

and, on the latter to take care of all essential, non-provable, truth:

“...what we can compute is what computer science is all about”.

However, as we argue later, Fortnow's faith in a classical Church-Turing Thesis that ensures:

“...Turing machines capture everything we can compute”,

may be as misplaced as his faith in the infallibility of standard interpretations of classical mathematics.

---

<sup>3</sup> Cf. Lance Fortnow's Web Log <[http://fortnow.com/lance/complog/archive/2003\\_04\\_06\\_archive.html](http://fortnow.com/lance/complog/archive/2003_04_06_archive.html)>.

The reason: There are, *prima facie*, reasonably strong arguments for a Kuhnian paradigm shift; not, as Wegner and Goldin believe, in the notion of computational problem solving, but in the standard interpretations of classical mathematical concepts.

However, Wegner and Goldin could be right in arguing that the direction of such a shift must be towards the incorporation of non-algorithmic effective methods into classical mathematical theory; presuming, from the following remarks, that this is, indeed, what “external interactions” are assumed to provide beyond classical Turing-computability:

(vi) ...that Turing machine models could completely describe all forms of computation ... contradicted Turing’s assertion that Turing machines could only formalize algorithmic problem solving ... and became a dogmatic principle of the theory of computation.

(vii) ...interaction between the program and the world (environment) that takes place during the computation plays a key role that cannot be replaced by any set of inputs determined prior to the computation.

(viii) ...a theory of concurrency and interaction requires a new conceptual framework, not just a refinement of what we find natural for sequential [algorithmic] computing.

(ix) ...the assumption that all of computation can be algorithmically specified is still widely accepted.

A widespread notion of particular interest, which seems to be recurrently implicit in Wegner and Goldin’s assertions too, is that mathematics is a dispensable tool of science, rather than its indispensable mother tongue.

However, the roots of such beliefs may also lie in ambiguities, in the classical definitions of foundational elements, that allow the introduction of non-constructive - hence non-

verifiable, non-computational, ambiguous, and essentially Platonic - elements into the standard interpretations of classical mathematics.

Consequently, standard interpretations of classical theory may, inadvertently, be weakening a desirable perception - of mathematics as the lingua franca of scientific expression - by ignoring the possibility that, since mathematics is, indeed, indisputably accepted as the language that most effectively expresses and communicates intuitive truth, the chasm between formal truth and provability must, of necessity, be bridgeable.<sup>4</sup>

We now consider one such bridge.

## 2. Non-constructivity in classical theory

We note that, in [Me90], Mendelson's following remarks (*italicised parenthetical qualifications added*) implicitly imply that classical definitions of various foundational elements can be argued as being either ambiguous, or non-constructive, or both:

"Here is the main conclusion I wish to draw: it is completely unwarranted to say that CT (*Church's Thesis*) is unprovable just because it states an equivalence between a vague, imprecise notion (effectively computable function) and a precise mathematical notion (partial-recursive function). ... The concepts and assumptions that support the notion of partial-recursive function are, in an essential way, no less vague and imprecise (*non-constructive, and intuitionistically objectionable*) than the notion of effectively computable function; the former are just more familiar and are part of a respectable theory with connections to other parts of logic and mathematics. (The notion of effectively computable function could have been incorporated into an axiomatic presentation of classical mathematics, but the acceptance of CT made this

---

<sup>4</sup> Of interest is Davis' argument [Da95] that an unprovable truth may, indeed, be arrived at algorithmically.

unnecessary.) ... Functions are defined in terms of sets, but the concept of set is no clearer (*not more non-constructive, and intuitionistically objectionable*) than that of function and a foundation of mathematics can be based on a theory using function as primitive notion instead of set. Tarski's definition of truth is formulated in set-theoretic terms, but the notion of set is no clearer (*not more non-constructive, and intuitionistically objectionable*), than that of truth. The model-theoretic definition of logical validity is based ultimately on set theory, the foundations of which are no clearer (*not more non-constructive, and intuitionistically objectionable*) than our intuitive (*non-constructive, and intuitionistically objectionable*) understanding of logical validity. ... The notion of Turing-computable function is no clearer (*not more non-constructive, and intuitionistically objectionable*) than, nor more mathematically useful (foundationally speaking) than, the notion of an effectively computable function."<sup>5</sup>

### 3. Constructive definitions

We further note that, if we accept that there can be non-algorithmic effective methods that are applicable instantiationally, but not algorithmically, then we can constructively express some of these classical notions<sup>6</sup> of various foundational elements such as “mathematical object”, “effective computability”, “truth of a formula under an interpretation”, “set”, “Church's Thesis” etc., in terms of a smaller number of primitive - formally undefined but intuitively well-accepted as unambiguous - mathematical terms as below:

---

<sup>5</sup> Reproduced from Bringsjord [Br93].

<sup>6</sup> We take Mendelson [Me64] as a standard exposition of classical theory and its standard interpretations.

(i) **Primitive mathematical object:** A primitive mathematical object of a formal mathematical language  $L$  is any symbol for an individual constant, predicate letter, or a function letter, which is defined as a primitive symbol of  $L$ .

(ii) **Formal mathematical object:** A formal mathematical object of a formal mathematical language  $L$  is any symbol for an individual constant, predicate letter, or a function letter that is either a primitive mathematical object of  $L$ , or that can be introduced through definition into  $L$  without inviting inconsistency.

(iii) **Mathematical object:** A mathematical object of a formal mathematical language  $L$  is any symbol that is either a primitive mathematical object, or a formal mathematical object of  $L$ .

(iv) **Set:** A set is the range of any function whose function letter is a mathematical object of a formal mathematical language  $L$ .

(v) **Instantiational computability:** A number-theoretic function  $F(x)$  is instantiationally computable if, and only if, given any natural number  $k$ , there is an effective method (which may depend on the value  $k$ ) to compute  $F(k)$ .

(vi) **Algorithmic computability:** A number-theoretic function  $F(x)$  is algorithmically computable if, and only if, there is an effective method (necessarily independent of  $x$ ) such that, given any natural number  $k$ , it can compute  $F(k)$ .

(vii) **Effective computability:** A number-theoretic function is effectively computable if, and only if, it is either instantiationally computable, or it is algorithmically computable.

(viii) **Instantiational truth:** A string  $[F(x)]$ <sup>7</sup> of a formal system P is instantiationally true under an interpretation M of P if, and only if, given any value  $k$  in M, there is an effective method (which may depend on the value  $k$ ) to determine that the interpreted proposition  $F(k)$  is satisfied in M ([Me64], p49-52).

(ix) **Algorithmic truth:** A string  $[F(x)]$  of a formal system P is algorithmically true under an interpretation M of P if, and only if, there is an effective method (necessarily independent of  $x$ ) such that, given any value  $k$  in M, it can determine that the interpreted proposition  $F(k)$  is satisfied in M.

(x) **Effective truth:** A string  $[F(x)]$  of a formal system P is effectively true under an interpretation M of P if, and only if, it is either instantiationally true in M, or it is algorithmically true in M.

(xi) **Instantiational Church Thesis:** If, for a given relation  $R(x)$ , and any element  $k$  in some interpretation M of a Peano Arithmetic, PA, there is an effective method such that it will determine whether  $R(k)$  holds in M or not, then every element of the domain D of M is the interpretation of some term of PA, and there is some PA-formula  $[R'(x)]$  such that:

$R(k)$  holds in M if, and only if,  $[R'(k)]$  is PA-provable.

(In other words, the Instantiational Church Thesis postulates that, if a relation  $R$  is effectively decidable instantiationally - possibly non-algorithmically - in an interpretation M of a Peano Arithmetic, PA, then  $R$  is expressible in PA, and the domain of  $R$  necessarily consists of only mathematical objects of PA, even if the predicate letter  $R$  is not, itself, a mathematical object of PA.)

---

<sup>7</sup> We use square brackets to distinguish between the uninterpreted string  $[F]$  of a formal system, and the symbolic expression “ $F$ ” that corresponds to it under a given interpretation that unambiguously assigns formal, or intuitive, meanings to each individual symbol of the expression “ $F$ ”.

(*xiii*) **Arithmetical Provability Thesis:** If, in some interpretation  $M$  of a Peano Arithmetic,  $PA$ , there is an effective method such that, for a given, total, arithmetical relation  $R(x)$ , and any element  $k$  in  $M$ , it will always determine that  $R(k)$  holds in  $M$ , then:

$[R(x)]$  is  $PA$ -provable.

(The Arithmetical Provability Thesis postulates that, if a total arithmetical relation  $R$  is effectively decidable algorithmically as always true in an interpretation  $M$  of a Peano Arithmetic  $PA$ , then the algorithm can be converted into a proof sequence for  $[R(x)]$  in  $PA$ .)

#### 4. Effective solvability of the Halting problem

The significance of defining classical foundational concepts in a constructive, and, *prima facie*, in an intuitionistically unobjectionable way, of weakening Church's Thesis to an equivalence (from an identity), and of supplementing it with an Arithmetical Provability Thesis, is seen in the following argument.

**Theorem 1:** The Arithmetical Provability Thesis implies that the Halting problem is effectively solvable.

**Proof:** If a number-theoretic function  $F(x)$  is Turing-computable, then it is partial recursive ([Me64], p233, Corollary 5.13). We may thus assume that such an  $F$  is obtained from a recursive function  $G$  by means of the unrestricted  $\mu$ -operator; in other words, that (cf. [Me64], p214):

$$F(x) = \mu y(G(x, y) = 0).$$

If  $[H(x, y)]$  expresses  $\sim(G(x, y) = 0)$  in a formal system of Arithmetic such as standard PA, we then consider the PA-provability, and truth in the standard interpretation  $M$  of PA, of the formula  $[H(a, y)]$  for a given numeral  $[a]$  of PA, as below:

(a) Let Q1 be the meta-assertion that there is no effective method in  $M$  to determine that, for any given  $b$  in  $M$ ,  $[H(a, b)]$  is satisfied in  $M$ . It follows that there is no algorithm in  $M$  to determine that, for any given  $b$  in  $M$ ,  $[H(a, b)]$  is satisfied in  $M$ .

Since  $G(a, y)$  is recursive, it follows that there is some finite  $k$  such that any Turing machine  $T1(y)$  that computes  $G(a, y)$  will halt and return the value 0 for  $y = k$ .

(b) Next, let Q2 be the meta-assertion that, for any given  $b$  in  $M$ , there is always an effective method - which may depend on  $b$  - that will determine that  $[H(a, b)]$  is satisfied in  $M$ , but that there is no algorithm in  $M$  to determine that, for any given  $b$  in  $M$ ,  $[H(a, b)]$  is satisfied in  $M$ .

Since  $G(a, y)$  is recursive, it follows that there is some finite  $k$  such that the  $T1(y)$  will halt, and return a symbol for self-termination (looping<sup>8</sup>) for  $y = k$ .

(c) Finally, let Q3 be the meta-assertion that there is an effective method in  $M$  that will determine, for any given  $b$  in  $M$ , that  $[H(a, b)]$  is satisfied in  $M$ . We then have that that  $[H(a, y)]$  is algorithmically true in  $M$ .

If we assume an Arithmetical Provability Thesis, it then follows that  $[H(a, y)]$  is PA-provable. Let  $h$  be the Gödel-number of  $[H(a, y)]$ .

---

<sup>8</sup> We note that any Turing machine can be designed to recognise a “looping” situation; it simply records every instantaneous tape description at the execution of each machine instruction, and compares the current instantaneous tape description with the record. It can thus be meta-programmed to abort a loop, and return a meta-symbol indicating self-termination.

We consider, now, Gödel's primitive recursive number-theoretic relation  $xBy$  ([Go31a], p22, definition 45), which holds in  $M$  if, and only if,  $x$  is the Gödel-number of a proof sequence in PA for the PA-formula whose Gödel-number is  $y$ .

It follows that there is some finite  $k$  such that any Turing machine  $T2(y)$ , which computes the characteristic function of  $xBh$ , will halt and return the value 0 for  $x = k$ .

Since  $Q1$ ,  $Q2$ , and  $Q3$  are mutually exclusive and exhaustive, it follows that, when run simultaneously over the sequence 1, 2, 3, ... of values for  $y$ , one of  $\{T1(y) // T2(y)\}$  will always halt for some finite value of  $y$  for any given  $a$ .

Thus, the Halting problem is effectively solvable if we assume an Arithmetical Provability Thesis.  $\square$

**Corollary 1.1:** The Arithmetical Provability Thesis implies that the parallel duo of Turing machines  $\{T1(y) // T2(y)\}$  is not a Turing machine.

**Corollary 1.2:** The Arithmetical Provability Thesis implies that the classical Church-Turing thesis is false.

## 5. Is there a case for an Arithmetical Provability Thesis?

The question of whether, as Fortnow believes, Turing machines can “capture everything we can compute”, or whether, as Wegner and Goldin suggest, they are “inappropriate as a universal foundation for computational problem solving”, could, thus, be answered decisively if there were, indeed, a case for introducing non-algorithmic effective methods into classical mathematics, and for introducing constructive foundational concepts such as that of an Arithmetical Provability Thesis.

## References

- [Br93] Bringsjord, S. 1993. *The Narrational Case Against Church's Thesis*. Easter APA meetings, Atlanta.  
 <Web page: <http://www.rpi.edu/~brings/SELPAP/CT/ct/ct.html>>
- [Da95] Davis, M. (1995). *Is mathematical insight algorithmic?* Behavioral and Brain Sciences, 13 (4), 659-60.  
 <PDF file: <http://citeseer.nj.nec.com/davis90is.html>>
- [Go31a] Gödel, Kurt. 1931. *On formally undecidable propositions of Principia Mathematica and related systems I*. Translated by Elliott Mendelson. In M. Davis (ed.). 1965. *The Undecidable*. Raven Press, New York.
- [Go31b] Gödel, Kurt. 1931. *On formally undecidable propositions of Principia Mathematica and related systems I*. Translated by B. Meltzer.  
 <Web version: <http://home.ddc.net/ygg/etext/godel/index.htm>>
- [Me64] Mendelson, Elliott. 1964. *Introduction to Mathematical Logic*. Van Norstrand, Princeton.
- [Me90] Mendelson, E. 1990. *Second Thoughts About Church's Thesis and Mathematical Proofs*. *Journal of Philosophy* **87.5**.
- [Tu36] Turing, Alan. 1936. *On computable numbers, with an application to the Entscheidungsproblem*.  
 <Web page: <http://www.abelard.org/turpap2/tp2-ie.asp> - index>

[WG03] Wegner, Peter and Goldin, Dina. 2003. *Computation Beyond Turing Machines*.  
Communications of the ACM, 46 (4) 2003.

<Peter Wegner's web page: <http://www.cs.brown.edu/people/pw/home.html> >

<Dina Goldin's web page: <http://www.cs.brown.edu/people/dqg> >