A foundational argument for defining effective computability formally and weakening the Church and Turing Theses

Bhupinder Singh Anand

Draft of April 16, 2012. An earlier version of this manuscript is arXived here.

Abstract

We conclude from Gödel’s Theorem VII of his seminal 1931 paper that every recursive function \( f(x_1, x_2) \) is representable in the first-order Peano Arithmetic PA by a formula \( [F(x_1, x_2, x_3)] \) which is algorithmically verifiable, but not algorithmically computable, if we assume that the negation of a universally quantified formula of the first-order predicate calculus is always indicative of the existence of a counter-example under the standard interpretation of PA. We conclude that the standard postulation of the Church-Turing Thesis does not hold if we define a number-theoretic formula as effectively computable if, and only if, it is algorithmically verifiable; and needs to be replaced by a weaker postulation of the Thesis as an equivalence.

Keywords Algorithmic computability, algorithmic verifiability, Aristotle’s particularisation, Church-Turing Thesis, effective computability, first-order, Gödel \( \beta \)-function, Peano Arithmetic PA, standard interpretation, Tarski, uniform method.

1 Background

1.1 The Computational Issue: Can Turing machines really capture everything we can compute?

In a short opinion paper, ‘Computation Beyond Turing Machines’ Peter Wegner and Dina Goldin advance 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.’

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

\[ \text{Subject class: LO; MSC: 03B10} \]

\[ \text{[WG93]} \]
(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 a dilemma with which not only Theoretical Computer Sciences, but all applied sciences that depend on mathematics—for providing a verifiable language to express their observations precisely—are faced:

Query 1 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 Lance Fortnow’s on-line rebuttalis 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’.


3Of interest is Martin Davis’ argument that an unprovable truth may, indeed, be arrived at ‘algorithmically’ [Davi95].
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.

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.

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.[8]

If so, we need to consider whether the roots of such interpretations may lie in removable ambiguities that currently persist in the classical definitions of foundational elements; ambiguities that allow the introduction of non-constructive—hence non-verifiable, non-computational, ambiguous, and essentially Platonic—elements into the standard interpretations of classical mathematics.

[8] We consider a possible such bridge in Appendix B.
1.2 The Philosophical Issue: Are axiomatic computational concepts really unambiguous?

This raises the corresponding philosophical question that is implicit in Selmer Bringsjord’s narrational case against Church’s Thesis:

Query 2 Is there a duality in the classical acceptance of non-constructive, foundational, concepts as axiomatic?

1.2.1 Mendelson’s thesis

We note that Elliott Mendelson is quoted there by Bringsjord as saying (italicised parenthetical qualifications added):

(i) “Here is the main conclusion I wish to draw: it is completely unwarranted to say that CT is unprovable just because it states an equivalence between a vague, imprecise notion (effectively computable function) and a precise mathematical notion (partial-recursive function)”.

(ii) “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 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”.

(iii) “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 . . .”

where:

(a) The Church-Turing Thesis, CT, is formulated as: “A function is effectively computable if and only if it is Turing-computable”.

6 [Me90].
(b) An effectively computable function is defined to be the computing of a function by an algorithm.

(c) The classical notion of an algorithm is expressed by Mendelson as: “...an effective and completely specified procedure for solving a whole class of problems. ... An algorithm does not require ingenuity; its application is prescribed in advance and does not depend upon any empirical or random factors”.

and, where Bringsjord paraphrases (iii) as:

(iv) “The notion of a formally defined program for guiding the operation of a TM is no clearer than, nor more mathematically useful (foundationally speaking) than, the notion of an algorithm”.

adding that:

(v) “This proposition, it would then seem, is the very heart of the matter. If (iv) is true then Mendelson has made his case; if this proposition is false, then his case is doomed, since we can chain back by modus tollens and negate (iii)”.

1.2.2 The concept of ‘constructive, and intuitionistically unobjectionable’

Now, prima facie, any formalisation of a ‘vague and imprecise’, ‘intuitive’ concept, say C, would normally be intended to capture the concept C, both faithfully and completely, within a constructive, and intuitionistically unobjectionable language L.

Clearly, we could disprove the thesis—that C and its formalisation L are interchangeable, hence equivalent—by showing that there is a constructive aspect of C that is formalisable in a constructive language L’, but that such formalisation cannot be assumed expressible in L without introducing inconsistency.

However, equally clearly, there can be no way of proving the equivalence, as this would contradict the premise that the concept is ‘vague and imprecise’, hence essentially open-ended in a non-definable way, and so non-formalisable.

Obviously, Mendelson’s assertion that there is no justification for claiming Church’s Thesis as unprovable must, therefore, rely on an interpretation that differs significantly from the above; for instance, his concept of provability may appeal to the axiomatic acceptability of ‘vague and imprecise’ concepts - as suggested by his remarks.

Now, we note that all the examples cited by Mendelson involve the decidability (computability) of an infinitude of meta-mathematical instances, where the distinction between the constructive meta-assertion that any given instance is individually decidable (computable), and the non-constructive meta-assertion

\[ \text{The terms ‘constructive’ and ‘constructive, and intuitionistically unobjectionable’ are used synonymously both in their familiar linguistic sense, and in a mathematically precise sense. Mathematically, we term a concept as ‘constructive, and intuitionistically unobjectionable’ if, and only if, it can be defined in terms of pre-existing concepts without inviting inconsistency. Otherwise, we understand it to mean unambiguously verifiable, by some ‘effective method’, within some finite, well-defined, language or meta-language. It may also be taken to correspond, broadly, to the concept of ‘constructive, and intuitionistically unobjectionable’ in the sense apparently intended by Gödel in his seminal 1931 paper [Go31], p.26.} \]
that all the instances are jointly decidable (computable) uniformly, is not ad-
dressed explicitly. However, §1.2.1(a), §1.2.1(b) and §1.2.1(c) appear to suggest
that Mendelson’s remarks relate implicitly to non-constructive meta-assertions.

Perhaps the real issue, then, is the one that emerges if we replace Mendelson’s
use of implicitly open-ended concepts such as ‘vague and imprecise’, and ‘intu-
itive’, by the more meta-mathematically meaningful concept of ‘non-constructive,
and intuitionistically objectionable’, as italicised and indicated parenthetically.

The essence of Mendelson meta-assertion §1.2.1(iii) then appears to be that,
if the classically accepted definitions of foundational concepts such as ‘partial
recursive function’, ‘function’, ‘Tarskian truth’ etc. are also non-constructive,
and intuitionistically objectionable, then replacing one non-constructive con-
cept by another may be psychologically unappealing, but it should be meta-
mathematically valid and acceptable.

1.2.3 The duality

Clearly, meta-assertion §1.2.1(iii) would stand refuted by a non-algorithmic ef-
fective method that is constructive. However, if it is explicitly—and, as sug-
gested by the nature of the arguments in Bringsjord’s paper, widely—accepted
at the outset that any effective method is necessarily algorithmic (i.e. uniform
as stated in §1.2.1(c) above), then any counter-argument to CT can, prima
facie, only offer non-algorithmic methods that may, paradoxically, be effective
intuitively in a non-constructive, and intuitionistically objectionable, way only!

Recognition of this dilemma is implicit in the admission that the various
arguments, as presented by Bringsjord in the case against Church’s Thesis—
including his narrational case—are open to reasonable, but inconclusive, refu-
tations.

Nevertheless, if we accept Mendelson’s thesis that the inter-changeability
of non-constructive concepts is valid in the foundations of mathematics, then
Bringsjord’s case against Church’s Thesis, since it is based similarly on non-
constructive concepts, should also be considered conclusive classically (even
though it cannot, prima facie, be considered constructively conclusive in an
intuitionistically unobjectionable way).

There is, thus, an apparent duality in the—seemingly extra-logical—decision
as to whether an argument based on non-constructive concepts may be accepted
as classically conclusive or not.

That this duality may originate in the very issues raised in Mendelson’s
remarks—concerning the non-constructive roots of foundational concepts that
are classically accepted as mathematically sound—is seen if we note that these
issues may be more significant than is, prima facie, apparent.

1.2.4 Definition of a formal mathematical object, and consequences

Thus, if we define a formal mathematical object as any symbol for an individ-
ual letter, function letter or a predicate letter that can be introduced through
definition into a formal theory without inviting inconsistency, then it can be ar-
gued that unrestricted, non-constructive, definitions of non-constructive, foun-
dational, set-theoretic concepts—such as ‘mapping’, ‘function’, ‘recursively emu-
meral set’, etc.—in terms of constructive number-theoretic concepts—such as
recursive number-theoretic functions and relations—may not always correspond to formal mathematical objects.

In other words, the assumption that every definition corresponds to a formal mathematical object may introduce a formal inconsistency into standard Peano Arithmetic and, ipso facto, into any Axiomatic Set Theory that models standard PA (which, loosely speaking, may be viewed as a constructive arithmetical parallel to Russell’s non-constructive impredicative set).

Since it can also be argued that the non-constructive element in Tarski’s definitions of ‘satisfiability’ and ‘truth’, and in Church’s Thesis, originate in a common, but removable, ambiguity in the interpretation of an effective method, perhaps it is worth considering whether Bringsjord’s acceptance of the assumption—that every constructive effective method is necessarily algorithmic, in the sense of being a uniform procedure as in §1.2.1(c) above—is mathematically necessary, or even whether it is at all intuitively tenable.

... uniform procedure, a property usually taken to be a necessary condition for a procedure to qualify as effective.

Thus, we may argue that we can explicitly and constructively define a non-algorithmic effective method as one that, in any given individual case, is individually effective if, and only if, it terminates finitely with a conclusive result; and an algorithmic effective method as one that is uniformly effective if, and only if, it terminates finitely, with a conclusive result, in any given individual case.

1.2.5 Bringsjord’s case against CT

Apropos the specific arguments against CT it would seem, prima facie, that an individually effective—even if not obviously constructive—method could be implicit in the following argument considered by Bringsjord:

“Assume for the sake of argument that all human cognition consists in the execution of effective processes (in brains, perhaps). It would then follow by CT that such processes are Turing-computable, i.e., that computationalism is true. However, if computationalism is false, while there remains incontrovertible evidence that human cognition consists in the execution of effective processes, CT is overthrown”.

Assuming computationalism is false, the issue in this argument would, then, be whether there is a constructive, and adequate, expression of human cognition in terms of individually effective methods.

An appeal to such an individually effective method may, in fact, be implicit in Bringsjord’s consideration of the predicate H, defined by:

\[ H(P, i) \text{ iff } (∃n)S(P, i, n) \]

\(^8\)We note that the possibility of a distinction between the interpreted number-theoretic meta-assertions, ‘For any given natural number \(x\), \(F(x)\) is true’ and ‘\(F(x)\) is true for all natural numbers \(x\)’, is not evident unless these are expressed symbolically as, ‘(\(∀x\))(\(F(x)\) is true)’ and ‘(\(∀x\))(\(F(x)\) is true)’, respectively. The issue, then, is whether the distinction can be given any mathematical significance. For instance, under a constructive formulation of Tarski’s definitions, we may qualify the latter by saying that it can be meaningfully asserted as a totality only if ‘\(F\)’ is a mathematical object.
where the predicate \( S(P, u, n) \) holds if, and only if, TM \( M \), running program \( P \) on input \( u \), halts in exactly \( n \) steps (= \( MP : u \Rightarrow n \) halt).

Bringsjord defines \( S \) as totally (and, implicitly, uniformly) computable in the sense that, given some triple \((P, u, n)\), there is some (uniform) program \( P^* \) which, running on some TM \( M^* \), can infallibly give us a verdict, Y (‘yes’) or N (‘no’), for whether or not \( S \) is true of this triple.

He then notes that, since the ability to (uniformly) determine, for a pair \((P, i)\), whether or not \( H \) is true of it, is equivalent to solving the full halting problem, \( H \) is not totally computable. However, he also notes that there is a program (implicitly non-uniform, and so, possibly, effective individually) which, when asked whether or not some TM \( M \) run by \( P \) on \( u \) halts, will produce Y iff \( MP : u \Rightarrow n \) halt. For this reason \( H \) is declared partially (implicitly, individually) computable.

More explicitly, Bringsjord remarks that Laszlo Kalmár’s refutation of CT\(^9\) is classically inconclusive mainly because it does not admit any uniform effective method, but appeals to the existence of an infinitude of individually effective methods.

However, Kalmár’s argument can be strengthened if we note, firstly, that we can extend the definition of classical Turing machines to include meta-routines that constructively self-terminate the computational process of a Turing machine whenever an instantaneous tape description repeats itself.

Secondly, under a reasonably plausible Arithmetically Provability Thesis\(^10\) that equates the formal PA provability of PA formulas with a constructive definition of the truth of their (sound) interpretations—in the structure \( N \) of the natural numbers—as evidenced by the computations of a Turing machine, it can be argued\(^11\) that the Halting problem can be solved, albeit non-algorithmically, and that the classical Church and Turing Theses do not hold.

Excepting that it always calculates Kalmár’s \( g(n) \) (defined below) constructively—even in the absence of a uniform procedure—within a fixed postulate system, the reasoning used in the above argument\(^12\) is, essentially, the same as Kalmár’s argument\(^13\) reproduced below from Bringsjord’s paper:

> “First, he draws our attention to a function \( g \) that isn’t Turing-computable, given that \( f \) \(^{14}\)

\[
g(x) = \mu y (f(x, y) = 0) = \text{the least } y \text{ such that } f(x, y) = 0
\]

if \( y \) exists; and 0 if there is no such \( y \)

Kalmár proceeds to point out that for any \( n \) in \( N \) for which a natural number \( y \) with \( f(n, y) = 0 \) exists, ‘an obvious method for the calculation of the least such \( y \) ... can be given,’ namely, calculate in succession the values \( f(n, 0), f(n, 1), f(n, 2), \ldots \) (which, by hypothesis, is something a computist or TM can do) until we hit a natural number \( m \) such that \( f(n, m) = 0 \), and set \( y = m \).

---

9. [Ka59].
10. Whose validity we address elsewhere in [An12].
11. Appendix B.
12. Appendix B.
13. [Ka59].
14. Bringsjord notes that the original proof can be found on page 741 of Kleene [Kl36].
On the other hand, for any natural number \( n \) for which we can prove, not in the frame of some fixed postulate system but by means of arbitrary—of course, correct—arguments that no natural number \( y \) with \( f(n, y) = 0 \) exists, we have also a method to calculate the value \( g(n) \) in a finite number of steps.

Kalmár goes on to argue as follows. The definition of \( g \) itself implies the tertium non datur, and from it and CT we can infer the existence of a natural number \( p \) which is such that

\[
(*) \text{ there is no natural number } y \text{ such that } f(p, y) = 0;
\]

and

\[
(**) \text{ this cannot be proved by any correct means.}
\]

Kalmár claims that (*) and (**) are very strange, and that therefore CT is at the very least implausible.”

We conclude that before we can cogently address 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’, we first need to address the philosophical question—implicit in Bringsjord—of whether, or not, the concept of ‘effective computability’ is capable of a constructive, and intuitionistically unobjectionable, definition; and the relation of such definition to that of formal provability.

We now consider the formal case for introduction of such a definition; and consider some consequences.

2 Introduction

We begin by noting that the following theses are classically equivalent:\(^{15}\)

**Standard Church’s Thesis**\(^ {16}\) A number-theoretic function (or relation, treated as a Boolean function) is effectively computable if, and only if, it is recursive\(^ {17}\).

**Standard Turing’s Thesis**\(^ {18}\) A number-theoretic function (or relation, treated as a Boolean function) is effectively computable if, and only if, it is Turing-computable\(^ {19}\).

In this paper we shall argue that, from a foundational perspective, the principle of Occam’s razor suggests the Theses should be postulated minimally as the following equivalences:

\(^{15}\) cf. [Me64], p.237.

\(^{16}\) Church’s (original) Thesis The effectively computable number-theoretic functions are the algorithmically computable number-theoretic functions [Ch36].

\(^{17}\) cf. [Me64], p.227.

\(^{18}\) After describing what he meant by “computable” numbers in the opening sentence of his 1936 paper on Computable Numbers [Tu36], Turing immediately expressed this thesis—albeit informally—as: “. . . the computable numbers include all numbers which could naturally be regarded as computable”.

\(^{19}\) cf. [BBJ03], p.33.
Weak Church’s Thesis A number-theoretic function (or relation, treated as a Boolean function) is effectively computable if, and only if, it is instantiationally equivalent to a recursive function (or relation, treated as a Boolean function).

Weak Turing’s Thesis A number-theoretic function (or relation, treated as a Boolean function) is effectively computable if, and only if, it is instantiationally equivalent to a Turing-computable function (or relation, treated as a Boolean function).

2.1 The need for explicitly distinguishing between ‘instantiational’ and ‘uniform’ methods

It is significant that both Kurt Gödel (initially) and Alonzo Church (subsequently—possibly under the influence of Gödel’s disquietude) enunciated Church’s formulation of ‘effective computability’ as a Thesis because Gödel was instinctively uncomfortable with accepting it as a definition that minimally captures the essence of ‘intuitive effective computability’.

Gödel’s reservations seem vindicated if we accept that a number-theoretic function can be effectively computable instantiationally (in the sense of being algorithmically verifiable as defined in Section 6 below), but not by a uniform method (in the sense of being algorithmically computable as defined in Section 6).

The significance of the fact (considered below in Section 7) that ‘truth’ too can be effectively decidable both instantiationally and by a uniform (algorithmic) method under the standard interpretation of PA is reflected in Gödel’s famous 1951 Gibbs lecture where he remarks:

“I wish to point out that one may conjecture the truth of a universal proposition (for example, that I shall be able to verify a certain property for any integer given to me) and at the same time conjecture that no general proof for this fact exists. It is easy to imagine situations in which both these conjectures would be very well founded. For the first half of it, this would, for example, be the case if the proposition in question were some equation \( F(n) = G(n) \) of two number-theoretical functions which could be verified up to very great numbers \( n \).”

Such a possibility is also implicit in Turing’s remark:

“The computable numbers do not include all (in the ordinary sense) definable numbers. Let \( P \) be a sequence whose \( n \)-th figure is 1 or 0 according as \( n \) is or is not satisfactory. It is an immediate consequence of the theorem of §8 that \( P \) is not computable. It is (so far as we know at present) possible that any assigned number of figures of \( P \) can be calculated, but not by a uniform process. When sufficiently

---

20 See [Go51].
21 [Tu36], §9(II), p.139.
22 Parikh’s paper [Pa71] can also be viewed as an attempt to investigate the consequences of expressing the essence of Gödel’s remarks formally.
23 [Tu36], §9(II), p.139.
many figures of P have been calculated, an essentially new method is necessary in order to obtain more figures.”

The need for placing such a distinction on a formal basis has also been expressed explicitly on occasion. Thus, Boolos, Burgess and Jeffrey define a diagonal function, $d$, any value of which can be decided effectively, although there is no single algorithm that can effectively compute $d$.

Now, the straightforward way of expressing this phenomenon should be to say that there are well-defined number-theoretic functions that are effectively computable instantiationally but not algorithmically. Yet, following Church and Turing, such functions are labeled as uncomputable.

“According to Turing’s Thesis, since $d$ is not Turing-computable, $d$ cannot be effectively computable. Why not? After all, although no Turing machine computes the function $d$, we were able to compute at least its first few values. For since, as we have noted, $f_1 = f_1 = f_1 = $ the empty function we have $d(1) = d(2) = d(3) = 1$. And it may seem that we can actually compute $d(n)$ for any positive integer $n$—if we don’t run out of time.”

The reluctance to treat a function such as $d(n)$—or the function $\Omega(n)$ that computes the $n$th digit in the decimal expression of a Chaitin constant—as computable, on the grounds that the ‘time’ needed to compute it increases monotonically with $n$, is curious; the same applies to any total Turing-computable function $f(n)$.

2.2 Distinguishing between algorithmically verifiability and algorithmic computability

We now show in Theorem 1 that if Aristotle’s particularisation is presumed valid over the structure $\mathbb{N}$ of the natural numbers—as is the case under the standard interpretation of the first-order Peano Arithmetic PA—then it follows from the instantiational nature of the (constructively defined) G"odel $\beta$-function that a primitive recursive relation can be instantiationally equivalent to an arithmetical relation, where the former is algorithmically computable over $\mathbb{N}$, whilst the latter is algorithmically verifiable but not algorithmically computable over $\mathbb{N}$.

---

24 Parikh’s distinction between ‘decidability’ and ‘feasibility’ in also appears to echo the need for such a distinction.
25 BBJ03, p. 37.
26 The issue here seems to be that, when using language to express the abstract objects of our individual, and common, mental ‘concept spaces’, we use the word ‘exists’ loosely in three senses, without making explicit distinctions between them (see An07c).
27 BBJ03, p.37.
28 Chaitin’s Halting Probability is given by $0 < \Omega = \sum 2^{-|p|} < 1$, where the summation is over all self-delimiting programs $p$ that halt, and $|p|$ is the size in bits of the halting program $p$; see Ct75.
29 The incongruity of this is addressed by Parikh in Pa71.
30 The only difference being that, in the latter case, we know there is a common ‘program’ of constant length that will compute $f(n)$ for any given natural number $n$; in the former, we know we may need distinctly different programs for computing $f(n)$ for different values of $n$, where the length of the program will, sometime, reference $n$.
31 See Definition 1 below.
32 By Kurt G"odel; see Appendix A, Section 8, Lemma 1.
Analogue distinctions in analysis: The distinction between algorithmically computable, and algorithmically verifiable but not algorithmically computable, number-theoretic functions seeks to reflect in arithmetic the essence of uniform method classically characterised by the distinctions in analysis between: (a) uniformly continuous, and point-wise continuous but not uniformly continuous, functions over an interval; (b) uniformly convergent, and point-wise convergent but not uniformly convergent, series.

A limitation of set theory and a possible barrier to computation: We note, further, that the above distinction cannot be reflected within a language—such as the set theory ZF—which identifies ‘equality’ with ‘equivalence’. Since functions are defined extensionally as mappings, such a language cannot recognise that a set which represents a primitive recursive function may be equivalent to, but computationally different from, a set that represents an arithmetical function; where the former function is algorithmically computable over \( \mathbb{N} \), whilst the latter is algorithmically verifiable but not algorithmically computable over \( \mathbb{N} \).

2.2.1 Significance of Gödel’s \( \beta \)-function

In Theorem VII of his seminal 1931 paper on formally undecidable arithmetical propositions, Gödel showed that, given a total number-theoretic function \( f(x) \) and any natural number \( n \), we can construct a primitive recursive function \( \beta(z, y, x) \) and natural numbers \( b_n, c_n \) such that \( \beta(b_n, c_n, i) = f(i) \) for all \( 0 \leq i \leq n \).

In this paper we shall essentially answer the following question affirmatively:

**Query 3** Does Gödel’s Theorem VII admit construction of an arithmetical function \( A(x) \) such that:

- (a) for any given natural number \( n \), there is an algorithm that can verify \( A(i) = f(i) \) for all \( 0 \leq i \leq n \) (hence \( A(x) \) may be said to be algorithmically verifiable if \( f(x) \) is recursive);
- (b) there is no algorithm that can verify \( A(i) = f(i) \) for all \( 0 \leq i \) (so \( A(x) \) may be said to be algorithmically uncomputable)?

2.2.2 Defining effective computability

We shall then formally define what it means for a formula of an arithmetical language to be:

(i) Algorithmically verifiable;
(ii) Algorithmically computable.

under an interpretation.

We shall further propose the definition:

**Effective computability:** A number-theoretic formula is effectively computable if, and only if, it is algorithmically verifiable.

---

\(^33\) See Section 2.1

\(^34\) \[Go31\], pp.30-31; reproduced below in Appendix A, Section 8.
Intuitionistically unobjectionable: We note first that since every finite set of integers is recursive, every well-defined number-theoretical formula is algorithmically verifiable, and so the above definition is intuitionistically unobjectionable; and second that the existence of an arithmetic formula that is algorithmically verifiable but not algorithmically computable (Theorem 1) supports Gødel’s reservations on Alonzo Church’s original intention to label his Thesis as a definition.

We shall then show that the algorithmically verifiable and the algorithmically computable PA formulas are well-defined under the standard interpretation of PA since:

(a) The PA-formulas are decidable as satisfied / unsatisfied or true / false under the standard interpretation of PA if, and only if, they are algorithmically verifiable;

(b) The algorithmically computable PA-formulas are a proper subset of the algorithmically verifiable PA-formulas;

(c) The PA-axioms are algorithmically computable as satisfied / true under the standard interpretation of PA;

(d) Generalisation and Modus Ponens preserve algorithmically computable truth under the standard interpretation of PA;

(e) The provable PA-formulas are precisely the ones that are algorithmically computable as satisfied / true under the standard interpretation of PA.

3 Comments, Notation and Standard Definitions

Comments We have taken some liberty in emphasising standard definitions selectively, and interspersing our arguments liberally with comments and references, generally of a foundational nature. These are intended to reflect our underlying thesis that essentially arithmetical problems appear more natural when expressed—and viewed—within the perspective of an interpretation of PA that appeals to the evidence provided by a deterministic Turing machine along the lines suggested in Section 7; a perspective that, by its very nature, cannot appeal implicitly to transfinite concepts.

Evidence “It is by now folklore . . . that one can view the values of a simple functional language as specifying evidence for propositions in a constructive logic . . . ”

Notation We use square brackets to indicate that the contents represent a symbol or a formula—of a formal theory—generally assumed to be well-formed unless otherwise indicated by the context. We use an asterisk to indicate that

35 See Section 2.1
36 A deterministic Turing machine has only one possible move from a given configuration.
37 [Mu91].
38 In other words, expressions inside the square brackets are to be only viewed syntactically as juxtaposition of symbols that are to be formed and manipulated upon strictly in accordance with specific rules for such formation and manipulation—in the manner of a mechanical or electronic device—without any regards to what the symbolism might represent semantically under an interpretation that gives them meaning.
the expression is to be interpreted semantically with respect to some well-defined interpretation.

**Definition 1 Aristotle’s particularisation** This holds that from an assertion such as:

‘It is not the case that: for any given \( x \), \( P^*(x) \) does not hold’, usually denoted symbolically by ‘\( \neg(\forall x)\neg P^*(x) \)’, we may always validly infer in the classical, Aristotelian, logic of predicates\(^{39}\) that:

‘There exists an unspecified \( x \) such that \( P^*(x) \) holds’, usually denoted symbolically by ‘\( (\exists x)P^*(x) \)’.

**The significance of Aristotle’s particularisation for the first-order predicate calculus:** We note that in a formal language the formula ‘\( (\exists x)P(x) \)’ is an abbreviation for the formula ‘\( \neg(\forall x)\neg P(x) \)’. The commonly accepted interpretation of this formula—and a fundamental tenet of classical logic unrestrictedly adopted as intuitively obvious by standard literature that seeks to build upon the formal first-order predicate calculus\(^{40}\)—tacitly appeals to Aristotelian particularisation.

However, L. E. J. Brouwer had noted in his seminal 1908 paper on the unreliability of logical principles\(^{41}\) that the commonly accepted interpretation of this formula is ambiguous if interpretation is intended over an infinite domain. He essentially argued that, even supposing the formula ‘\( P(x) \)’ of a formal Arithmetical language interprets as an arithmetical relation denoted by ‘\( P^*(x) \)’, and the formula ‘\( \neg(\forall x)\neg P(x) \)’ as the arithmetical proposition denoted by ‘\( \neg(\forall x)\neg P^*(x) \)’, the formula ‘\( (\exists x)P(x) \)’ need not interpret as the arithmetical proposition denoted by the usual abbreviation ‘\( (\exists x)P^*(x) \)’; and that such postulation is invalid as a general logical principle in the absence of a means for constructing some putative object \( a \) for which the proposition \( P^*(a) \) holds in the domain of the interpretation.

Hence we shall follow the convention that the assumption that ‘\( (\exists x)P^*(x) \)’ is the intended interpretation of the formula ‘\( (\exists x)P(x) \)’—which is essentially the assumption that Aristotle’s particularisation holds over the domain of the interpretation—must always be explicit.

**The significance of Aristotle’s particularisation for PA:** In order to avoid intuitionistic objections to his reasoning, Gödel introduced the syntactic property of \( \omega \)-consistency as an explicit assumption in his formal reasoning in his seminal 1931 paper on formally undecidable arithmetical propositions\(^{42}\). Gödel explained at some length\(^{43}\) that his reasons for introducing \( \omega \)-consistency explicitly was to avoid appealing to the semantic concept of classical arithmetical truth in Aristotle’s logic of predicates (which presumes Aristotle’s particularisation).

However, we note that the two concepts are meta-mathematically equivalent in the sense that, if PA is consistent, then PA is \( \omega \)-consistent if, and only if, Aristotle’s particularisation holds under the standard interpretation of PA.

---

\(^{39}\) HA28, pp.58-59.

\(^{40}\) See HA28, p.48; Sk25, p.515; Go31, p.32.; K39, p.160; Go33, p.90; BF53, p.46; Be59, pp.178 & 218; Sm63, p.3; Va63, pp.314-315; Qu63, pp.12-13; Kn65, p.60; Con66, p.4; Mu66, p.52(i); Nv64, p.92; Ekv64, p.53; Sm67, p.13; Ull65, p.xxv; Kn67, p.xvii; Be69, p.174; Mu69, p.18, Ex.3; BBJ93, p.102.

\(^{41}\) Be98.

\(^{42}\) Go31, p.23 and p.28.

\(^{43}\) In his introduction on p.9 of Go31.
Definition 2 The structure $\mathbb{N}$ The structure of the natural numbers—namely, 
$\{N \ (the \ set \ of \ natural \ numbers); \ = \ (equality); \ ' \ (the \ successor \ function); \ + \ (the \ addition \ function); \ * \ (the \ product \ function); \ 0 \ (the \ null \ element)\}.$

Definition 3 The axioms of first-order Peano Arithmetic (PA)

$PA_1 \ [x_1 = x_2) \rightarrow ((x_1 = x_3) \rightarrow (x_2 = x_3)]];$
$PA_2 \ [(x_1 = x_2) \rightarrow (x_1' = x_2')];$
$PA_3 \ [0 \neq x_1];$
$PA_4 \ [(x_1' = x_2') \rightarrow (x_1 = x_2)];$
$PA_5 \ [(x_1 + 0) = x_1];$
$PA_6 \ [(x_1 + x_2') = (x_1 + x_2)'];$
$PA_7 \ [(x_1 \cdot 0) = 0];$
$PA_8 \ [(x_1 \cdot x_2') = ((x_1 \cdot x_2) + x_1)];$
$PA_9 \ For \ any \ well-formed \ formula \ [F(x)] \ of \ PA:\$
$\ [F(0) \rightarrow (((\forall x)(F(x) \rightarrow F(x')))) \rightarrow (\forall x)F(x)]].$

Definition 4 Generalisation in PA If $[A]$ is PA-provable, then so is $[\forall x]A].$

Definition 5 Modus Ponens in PA If $[A]$ and $[A \rightarrow B]$ are PA-provable, then so is $[B].$

Definition 6 Standard interpretation of PA The standard interpretation of PA over the structure $\mathbb{N}$ is the one in which the logical constants have their ‘usual’ interpretation$^{44}$ in Aristotle’s logic of predicates (which subsumes Aristotle’s particularisation), and$^{45}$

(a) the set of non-negative integers is the domain;
(b) the symbol $[0]$ interprets as the integer 0;
(c) the symbol $['']$ interprets as the successor operation (addition of 1);
(d) the symbols $[+]$ and $[*]$ interpret as ordinary addition and multiplication;
(e) the symbol $=[]=] interprets as the identity relation.

Definition 7 Soundness (formal system - non-standard): A formal system $S$ is sound under an interpretation $\mathcal{I}_S$ with respect to a domain $\mathbb{D}$ if, and only if, every theorem $[T]$ of $S$ translates as $[T]$ is true under $\mathcal{I}_S$ in $\mathbb{D}.$

Definition 8 Soundness (interpretation - non-standard): An interpretation $\mathcal{I}_S$ of a formal system $S$ is sound with respect to a domain $\mathbb{D}$ if, and only if, $S$ is sound under the interpretation $\mathcal{I}_S$ over the domain $\mathbb{D}.$

Soundness in classical logic: In classical logic, a formal system $S$ is sometimes defined as ‘sound’ if, and only if, it has an interpretation; and an interpretation is defined as the assignment of meanings to the symbols, and truth-values to the sentences of the formal system. Moreover, any such interpretation is defined as a model$^{46}$ of the formal system. This definition suffers, however, from an implicit circularity: the formal logic $L$ underlying any interpretation of $S$ is implicitly assumed to be ‘sound’. The above definitions seek to avoid this implicit circularity by delinking the defined ‘soundness’ of a formal system under an interpretation from the implicit ‘soundness’ of the formal logic underlying the interpretation. This admits the case where, even if $L_1$ and $L_2$ are implicitly assumed to be sound, $S + L_1$ is sound, but $S + L_2$ is not. Moreover, an interpretation of $S$ is now a model for $S$ if, and only if, it is sound$^{47}$

---

44See [Meta], p.49.
45See [Meta], p.107.
46We follow the definition in [Meta], p.51.
47My thanks to Professor Rohit Parikh for highlighting the need for making such a distinction explicit.
4 Gödel’s Theorem VII and algorithmically verifiable, but not algorithmically computable, arithmetical propositions

In his seminal 1931 paper on formally undecidable arithmetical propositions, Gödel defined a curious primitive recursive function—Gödel’s $\beta$-function—as

\[
\beta(x_1, x_2, x_3) = rm(1 + (x_3 + 1) \times x_2, x_1)
\]

where $rm(x_1, x_2)$ denotes the remainder obtained on dividing $x_2$ by $x_1$.

Gödel showed that the above function has the remarkable property that:

**Lemma 1** For any given denumerable sequence of natural numbers, say $f(k,0)$, $f(k,1)$, ..., and any given natural number $n$, we can construct natural numbers $b,c,j$ such that:

(i) \[ j = \max(n, f(k,0), f(k,1), \ldots, f(k,n)) \]

(ii) \[ c = j! \]

(iii) \[ \beta(b,c,i) = f(k,i) \text{ for } 0 \leq i \leq n. \]

**Proof** This is a standard result.\[49^5\]


Gödel’s original argument: We reproduce Gödel’s original argument—which yields this critical lemma—in Appendix A, Section 8.

Now we have the standard definition\[50^6\]:

**Definition 10** A number-theoretic function $f(x_1, \ldots, x_n)$ is said to be representable in PA if, and only if, there is a PA formula $[F(x_1, \ldots, x_{n+1})]$ with the free variables $[x_1, \ldots, x_{n+1}]$, such that, for any given natural numbers $k_1, \ldots, k_{n+1}$:

(i) if $f(k_1, \ldots, k_n) = k_{n+1}$ then PA proves: $[F(k_1, \ldots, k_n, k_{n+1})]$;

(ii) PA proves: $[\exists x_{n+1} F(k_1, \ldots, k_n, x_{n+1})]$.

The function $f(x_1, \ldots, x_n)$ is said to be strongly representable in PA if we further have that:

(iii) PA proves: $[\exists_1 x_{n+1} F(x_1, \ldots, x_n, x_{n+1})]$

**Interpretation of ‘$\exists_1$’**: The symbol ‘$\exists_1$’ denotes ‘uniqueness’ under an interpretation which assumes that Aristotle’s particularisation holds in the domain of the interpretation. Formally, however, the PA formula $[(\exists_1 x_3)F(x_1, x_2, x_3)]$ is merely a short-hand notation for the PA formula $[\neg(\forall x_3)\neg F(x_1, x_2, x_3) \land (\forall y)(\forall z) (F(x_1, x_2, y) \land F(x_1, x_2, z) \rightarrow y = z)].$

We then have:

**Lemma 2** $\beta(x_1, x_2, x_3)$ is strongly represented in PA by $[Bt(x_1, x_2, x_3, x_4)]$, which is defined as follows:

---

\[48^6\] Cf. [Go31], p.31, Lemma 1; [Me64], p.131, Proposition 3.21.

\[49^5\] Cf. [Go31], p.31, Lemma 1; [Me64], p.131, Proposition 3.22.

\[50^6\] [Me64], p.118.
\[(\exists w)(x_1 = ((1 + (x_3 + 1) \cdot x_2) \cdot w + x_4) \land (x_4 < 1 + (x_1 + 1) \cdot x_2))\].

**Proof** This is a standard result.\(^{[54]}\)

Gödel further showed (also under the tacit, but critical, presumption of Aristotle’s particularisation\(^{[55]}\) that:

**Lemma 3** If \(f(x_1, x_2)\) is a recursive function defined by:

(i) \(f(x_1, 0) = g(x_1)\)

(ii) \(f(x_1, (x_2 + 1)) = h(x_1, x_2, f(x_1, x_2))\)

where \(g(x_1)\) and \(h(x_1, x_2, x_3)\) are recursive functions of lower rank\(^{[56]}\) that are represented in PA by well-formed formulas \([G(x_1, x_2)]\) and \([H(x_1, x_2, x_3, x_4)]\), then \(f(x_1, x_2)\) is represented in PA by the following well-formed formula, denoted by \([F(x_1, x_2, x_3)]\):

\[[(3u)(\exists v)((3w)(Bt(u, v, 0, w) \land G(x_1, w)) \land Bt(u, v, x_2, x_3) \land (\forall w)(w < x_2 \rightarrow (\exists y)(\exists z)(Bt(u, v, y) \land Bt(u, v, (w + 1), z) \land H(x_1, w, y, z))]].\]

**Proof** This is a standard result.\(^{[57]}\)

Gödel’s original argument: We reproduce Gödel’s original argument and proof of this critical lemma in Appendix A, Section 8.

### 4.1 What does “\((\exists x_3)F(k, m, x_3)\)” is provable” assert under the standard interpretation of PA?

Now, if the PA formula \([F(x_1, x_2, x_3)]\) represents in PA the recursive function denoted by \(f(x_1, x_2)\) then by definition, for any given numerals \([k], [m]\), the formula \([\exists x_3]F(k, m, x_3)]\) is provable in PA; and true under the standard interpretation of PA. We thus have that:

**Lemma 4** \("(\exists x_3)F(k, m, x_3)\) is true under the standard interpretation of PA” is the assertion that:

Given any natural numbers \(k, m\), we can construct natural numbers \(t_{(k,m)}, u_{(k,m)}, v_{(k,m)}\) —all functions of \(k, m\)—such that:

(a) \(\beta(u_{(k,m)}, v_{(k,m)}, 0) = g(k)\);

(b) for all \(i < m\) \(\beta(u_{(k,m)}, v_{(k,m)}, i) = h(k, i, f(k, i))\);

(c) \(\beta(u_{(k,m)}, v_{(k,m)}, m) = t_{(k,m)}\);

where \(f(x_1, x_2), g(x_1)\) and \(h(x_1, x_2, x_3)\) are any recursive functions that are formally represented in PA by \(F(x_1, x_2, x_3), G(x_1, x_2)\) and \(H(x_1, x_2, x_3, x_4)\) respectively such that:

\(^{[51]}\) cf. \(\text{Me64}, \) p.131, proposition 3.21.

\(^{[52]}\) The implicit assumption being that the negation of a universally quantified formula of the first-order predicate calculus is indicative of “the existence of a counter-example” —\(\text{Go31}, \) p.32.

\(^{[53]}\) cf. \(\text{Me64}, \) p.132; \(\text{Go31}, \) p.30(2).

\(^{[54]}\) cf. \(\text{Go31}, \) p.31(2); \(\text{Me64}, \) p.132.
Theorem 1

Under the standard interpretation of PA $[(∃_1 x_3) F(x_1, x_2, x_3)]$ is algorithmically verifiable, but not algorithmically computable, as always true over $\mathbb{N}$.

Proof

For any given natural numbers $k$ and $m$, if $[F(x_1, x_2, x_3)]$ interprets as a well-defined arithmetical relation under the standard interpretation of PA, then we can define a deterministic Turing machine $TM$ that can ‘construct’ the sequences $f(k, 0), f(k, 1), \ldots, f(k, m)$ and $β(u_{(k, m)}, v_{(k, m)}, 0), β(u_{(k, m)}, v_{(k, m)}, 1), \ldots, β(u_{(k, m)}, v_{(k, m)}, m)$ and give evidence to verify the assertion.

Does $[F(x_1, x_2, x_3)]$ interpret as a well-defined predicate? A critical issue that we do not address here is whether the PA formula $[F(x_1, x_2, x_3)]$ can be considered to interpret under a sound interpretation of PA as a well-defined predicate, since the denumerable sequences $\{f(k, 0), f(k, 1), \ldots, f(k, m), m_p : p > 0$ and $m_p$ is not equal to $m_q$ if $p$ is not equal to $q\}$—are represented by denumerable, distinctly different, functions $β(u_{p_1}, v_{p_2}, i)$ respectively. There are thus denumerable pairs $(u_{p_1}, v_{p_2})$ for which $β(u_{p_1}, v_{p_2}, i)$ yields the sequence $f(k, 0), f(k, 1), \ldots, f(k, m)$.

We now see that:

(i) $f(k, 0) = g(k)$
(ii) $f(k, (y + 1)) = h(k, y, f(k, y))$ for all $y < m$
(iii) $g(x_1)$ and $h(x_1, x_2, x_3)$ are recursive functions that are assumed to be of lower rank than $f(x_1, x_2)$.

Proof

For any given natural numbers $k$ and $m$, if $[F(x_1, x_2, x_3)]$ interprets as a well-defined arithmetical relation under the standard interpretation of PA, then we can define a deterministic Turing machine $TM$ that can ‘construct’ the sequences $f(k, 0), f(k, 1), \ldots, f(k, m)$ and $β(u_{(k, m)}, v_{(k, m)}, 0), β(u_{(k, m)}, v_{(k, m)}, 1), \ldots, β(u_{(k, m)}, v_{(k, m)}, m)$ and give evidence to verify the assertion.

$[(∃_1 x_3) F(x_1, x_2, x_3)]$ is PA-provable for any given numerals $[k, m]$.

Hence $[(∃_1 x_3) F(k, m, x_3)]$ is true under the standard interpretation of PA. It then follows from the definition of $[F(x_1, x_2, x_3)]$ in Lemma 3 that, for any given natural numbers $k, m$, we can construct some pair of natural numbers $u_{(k, m)}, v_{(k, m)}$—where $u_{(k, m)}, v_{(k, m)}$ are functions of the given natural numbers $k$ and $m$—such that:

(a) $β(u_{(k, m)}, v_{(k, m)}, i) = f(k, i)$ for $0 \leq i \leq m$;
(b) $F^*(k, m, f(k, m))$ holds in $\mathbb{N}$.

Since $β(x_1, x_2, x_3)$ is primitive recursive, $β(u_{(k, m)}, v_{(k, m)}, i)$ defines a deterministic Turing machine $TM$ that can ‘construct’ the denumerable sequence $f'(k, 0), f'(k, 1), \ldots$ for any given natural numbers $k$ and $m$ such that:

(c) $f(k, i) = f'(k, i)$ for $0 \leq i \leq m$.

We can thus define a deterministic Turing machine $TM$ that will give evidence that the PA formula $[(∃_1 x_3) F(k, m, x_3)]$ is true under the standard interpretation of PA.

Hence $[(∃_1 x_3) F(x_1, x_2, x_3)]$ is algorithmically verifiable over $\mathbb{N}$ under the standard interpretation of PA.
(2) Now, the pair of natural numbers \( u(x_1,x_2), v(x_1,x_2) \) are defined such that:

(a) \( \beta(u(x_1,x_2), v(x_1,x_2), i) = f(x_1, i) \) for \( 0 \leq i \leq x_2 \);

(b) \( F^*(x_1,x_2, f(x_1,x_2)) \) holds in \( \mathbb{N} \);

where \( v(x_1,x_2) \) is defined in Lemma 3 as \( j! \), and:

(c) \( j = \max(n, f(x_1,0), f(x_1,1), \ldots, f(x_1,x_2)) \);

(d) \( n \) is the ‘number’ of terms in the sequence \( f(x_1,0), f(x_1,1), \ldots, f(x_1,x_2) \).

Since \( j \) is not definable for a denumerable sequence \( \beta(u(x_1,x_2), v(x_1,x_2), i) \) we cannot define a denumerable sequence \( f'(x_1,0), f'(x_1,1), \ldots \) such that:

(e) \( f(k,i) = f'(k,i) \) for all \( i \geq 0 \).

We cannot thus define a deterministic Turing machine \( TM \) that will give evidence that the PA formula \( (\exists x_3) F(x_1,x_2,x_3) \) interprets as true under the standard interpretation of PA for any given sequence of numerals \( [(a_1,a_2)] \).

Hence \( (\exists x_3) F(x_1,x_2,x_3) \) is not algorithmically computable over \( \mathbb{N} \) under the standard interpretation of PA.

The theorem follows.

\[ \square \]

**Corollary 1** If the standard interpretation of PA is sound, then the classical Church and Turing theses are false.\(^{55}\)

The above theorem now suggests the following definition:

**Definition 11** **Effective computability:** A number-theoretic function is effectively computable if, and only if, it is algorithmically verifiable.

Such a definition of effective computability now allows the classical Church and Turing theses to be expressed as the weak equivalences in §2—rather than as identities—without any apparent loss of generality.

However, we still need to formally define what it means for a number-theoretic formula of an arithmetical language to be:

(i) Algorithmically verifiable;

(ii) Algorithmically computable.

under an interpretation.

\(^{55}\)See also Corollary §4.
5 Interpretation of an arithmetical language in terms of Turing computability

We begin by noting that we can, in principle, define the classical ‘satisfaction’ and ‘truth’ of the formulas of a first order arithmetical language, such as PA, verifiably under an interpretation using as evidence the computations of a simple functional language.

Such definitions follow straightforwardly for the atomic formulas of the language (i.e., those without the logical constants that correspond to ‘negation’, ‘conjunction’, ‘implication’ and ‘quantification’) from the standard definition of a deterministic Turing machine based essentially on Alan Turing’s seminal 1936 paper on computable numbers.

Moreover, it follows from Alfred Tarski’s seminal 1933 paper on the concept of truth in the languages of the deductive sciences that the ‘satisfaction’ and ‘truth’ of those formulas of a first-order language which contain logical constants can be inductively defined, under an interpretation, in terms of the ‘satisfaction’ and ‘truth’ of the interpretations of only the atomic formulas of the language.

Hence the ‘satisfaction’ and ‘truth’ of those formulas (of an arithmetical language) which contain logical constants can, in principle, also be defined verifiably under an interpretation using as evidence the computations of a deterministic Turing machine.

We show in Section 7 that this is indeed the case for PA under the standard interpretation \( \mathcal{I}_{PA}[N, \text{Standard}] \), when this is explicitly defined as in Section 7.1.

We show in Section 6, moreover, that we can further define ‘algorithmic truth’ and ‘algorithmic falsehood’ under \( \mathcal{I}_{PA}[N, \text{Standard}] \) such that the PA axioms interpret as always algorithmically true.

Significance of ‘algorithmic truth’: The algorithmically true propositions of \( N \) under \( \mathcal{I}_{PA}[N, \text{Standard}] \) are thus a proper subset of the verifiably true propositions of \( N \) under \( \mathcal{I}_{PA}[N, \text{Standard}] \); and suggest a possible finitary ‘model’ of PA that would establish the consistency of PA constructively.

5.1 The definitions of ‘algorithmic truth’ and ‘algorithmic falsehood’ under \( \mathcal{I}_{PA}[N, \text{Standard}] \) are not symmetric with respect to ‘truth’ and ‘falsehood’ under \( \mathcal{I}_{PA}[N, \text{Standard}] \)

However, the definitions (in Section 6) of ‘algorithmic truth’ and ‘algorithmic falsehood’ under \( \mathcal{I}_{PA}[N, \text{Standard}] \) are not symmetric with respect to classical (verifiable) ‘truth’ and ‘falsehood’ under \( \mathcal{I}_{PA}[N, \text{Standard}] \).

For instance, if a formula \( [F(x_1, x_2, \ldots, x_n)] \) of an arithmetic is algorithmically true under an interpretation (such as \( \mathcal{I}_{PA}[N, \text{Standard}] \)) that appeals to the evidence provided by the computations of a deterministic Turing machine, then, for any given denumerable sequence of numerical values \( [a_1, a_2, \ldots, a_n] \), the formula \( [F(a_1, a_2, \ldots, a_n)] \) is also algorithmically true under the interpretation.

\[ \text{References:} \]
\[ \begin{align*}
\text{[Me64]}, \text{pp.229-231.} \\
\text{[Tu36].} \\
\text{[Ta33].} \\
\text{[Me64], p.51.} \\
\end{align*} \]
Denumerable sequence: We shall presume that any such sequence is ‘given’ in the sense of being defined by the ‘evidence’ of a deterministic Turing machine.

In other words, there is a deterministic Turing machine which can provide evidence that the interpretation $F^*(a_1, a_2, \ldots, a_n)$ of $[F(a_1, a_2, \ldots, a_n)]$ holds in $\mathbb{N}$ for any given denumerable sequence of natural numbers $(a_1, a_2, \ldots)$.

**Defining the term ‘hold’**: We define the term ‘hold’—when used in connection with an interpretation of a formal language and, more specifically, with reference to the operations of a deterministic Turing machine associated with the atomic formulas of the language—explicitly in Section 7, the aim being to avoid appealing to the classically subjective (and existential) connotation implicitly associated with the term under an implicitly defined standard interpretation of an arithmetic.$^{60}$

However, if a formula $[F(x_1, x_2, \ldots, x_n)]$ of an arithmetic is algorithmically false under an interpretation that appeals to the evidence provided by the computations of a deterministic Turing machine, we cannot conclude that there is a denumerable sequence of numerical values $[a_1, a_2, \ldots]$ such that the formula $[F(a_1, a_2, \ldots, a_n)]$ is algorithmically false under the interpretation.

Reason: If a formula $[F(x_1, x_2, \ldots, x_n)]$ of an arithmetic is algorithmically false under such an interpretation, then we can only conclude that there is no deterministic Turing machine which can provide evidence that the interpretation $F^*(x_1, x_2, \ldots, x_n)$ holds in $\mathbb{N}$ for any given denumerable sequence of natural numbers $(a_1, a_2, \ldots)$; we cannot conclude that there is a denumerable sequence of natural numbers $(b_1, b_2, \ldots)$ such that $F^*(b_1, b_2, \ldots, b_n)$ does not hold in $\mathbb{N}$.

Such a conclusion would require:

(i) either some additional evidence that will verify for some assignment of numerical values to the free variables of $[F]$ that the corresponding interpretation $F^*$ does not hold$^{61}$;

(ii) or the additional assumption that either Aristotle’s particularisation holds over the domain of the interpretation (as is implicitly presumed under the standard interpretation of PA) or that the arithmetic is $\omega$-consistent$^{62}$.

**An issue of consistency**: An issue that we do not address here is whether the assumption of Aristotle’s particularisation (or that of $\omega$-completeness) is consistent with the evidence provided by the computations of a deterministic Turing machine for defining the satisfaction and truth of the formulas of an arithmetic under an interpretation.

### 6 Defining algorithmic verifiability and algorithmic computability

The asymmetry of Section 5.1 suggests the following two concepts.$^{63}$

---

60 As, for instance, in [Go31].

61 Essentially reflecting Brouwer’s objection to the assumption of Aristotle’s particularisation over an infinite domain.

62 An assumption explicitly introduced by Gödel in [Go31].

63 My thanks to Dr. Chaitanya H. Mehta for advising that the focus of this investigation should be the distinction between these two concepts.
Definition 12 \textit{Algorithmic verifiability:}\n
An arithmetical formula \([\forall x]F(x)\) is algorithmically verifiable as true under an interpretation if, and only if, for any given numeral \([a]\), we can define a deterministic Turing machine \(TM\) that computes \([F(a)]\) in the Tarskian sense (as defined in Section 7) such that \(TM\) will halt on null input if, and only if, \([F(a)]\) interprets as true under the interpretation.

Tarskian interpretation of an arithmetical language \textit{verifiably} in terms of Turing computability We show in Section 7 that the ‘algorithmic verifiability’ of the formulas of a formal language which contain logical constants can be inductively defined under an interpretation in terms of the ‘algorithmic verifiability’ of the interpretations of the atomic formulas of the language; further, that the PA-formulas are decidable under the standard interpretation of PA if, and only if, they are algorithmically verifiable under the interpretation (Corollary ??).

Definition 13 \textit{Algorithmic computability:}\n
An arithmetical formula \([\forall x]F(x)\) is algorithmically computable as true under an interpretation if, and only if, we can define a deterministic Turing machine \(TM\) that computes \([F(x)]\) in the Tarskian sense (as defined in Section 7) such that, for any given numeral \([a]\), \(TM\) will halt on \([a]\) if, and only if, \([F(a)]\) interprets as true under the interpretation.

Tarskian interpretation of an arithmetical language \textit{algorithmically} in terms of Turing computability We show in Section 7 that the ‘algorithmic computability’ of the formulas of a formal language which contain logical constants can also be inductively defined under an interpretation in terms of the ‘algorithmic computability’ of the interpretations of the atomic formulas of the language; further, that the PA-formulas are decidable under an algorithmic interpretation of PA if, and only if, they are algorithmically computable under the interpretation.

We now show that the above concepts are well-defined under the standard interpretation of PA.

7 Standard definitions of the ‘satisfiability’ and ‘truth’ of a formal language under an interpretation

We note that standard interpretations of the formal reasoning and conclusions of classical first order theory—based primarily on the work of Cantor, Gödel, Tarski, and Turing—seem to admit the impression that the classical, Tarskian, truth (satisfiability) of the propositions of a formal mathematical language under an interpretation is, both, non-algorithmic and essentially unverifiable constructively.\(^{64}\)

However—if mathematics is to serve as a universal set of languages of, both, precise expression and unambiguous communication—such interpretations may need to be balanced by an alternative, constructive and intuitionistically unobjectionable, interpretation—of classical foundational concepts—in which non-algorithmic truth (satisfiability) is defined effectively.\(^{65}\)

For instance, we note that—essentially following standard expositions\(^{66}\) of Tarski’s inductive definitions on the ‘satisfiability’ and ‘truth’ of the formulas of a formal language under an interpretation—we can define:

\(^{64}\)Thus giving rise to J. R. Lucas’ Gödelian Argument in \cite{Lu61}; but see also \cite{An07a, An07b, An07c} and \cite{An08}.

\(^{65}\)Such an interpretation is broadly outlined in \cite{An07d}.

\(^{66}\)cf. \cite{Me63}, p.51.
Definition 14 If \([A] \) is an atomic formula \([A(x_1, x_2, \ldots, x_n)] \) of a formal language \(S\), then the denumerable sequence \((a_1, a_2, \ldots)\) in the domain \(D\) of an interpretation \(\mathcal{I}_{S(D)}\) of \(S\) satisfies \([A]\) if, and only if:

(i) \([A(x_1, x_2, \ldots, x_n)]\) interprets under \(\mathcal{I}_{S(D)}\) as a relation \(A^*(x_1, x_2, \ldots, x_n)\) in \(D\) for a witness \(W_D\) of \(D\):

(ii) there is a Satisfaction Method, \(SM(\mathcal{I}_{S(D)})\) that provides objective evidence\(^{67}\) by which any witness \(W_D\) of \(D\) can define for any atomic formula \([A(x_1, x_2, \ldots, x_n)]\) of \(S\), and any given denumerable sequence \((b_1, b_2, \ldots)\) of \(D\), whether the proposition \(A^*(b_1, b_2, \ldots, b_n)\) holds or not in \(D\);

(iii) \(A^*(a_1, a_2, \ldots, a_n)\) holds in \(D\) for \(W_D\).

Witness: From a constructive perspective, the existence of a ‘witness’ as in (i) above is implicit in the usual expositions of Tarski’s definitions.

Satisfaction Method: From a constructive perspective, the existence of a Satisfaction Method as in (ii) above is also implicit in the usual expositions of Tarski’s definitions.

A constructive perspective: We highlight the word ‘define’ in (ii) above to emphasise the constructive perspective underlying this paper; which is that the concepts of ‘satisfaction’ and ‘truth’ under an interpretation are to be explicitly viewed as objective assignments by a convention that is witness-independent. A Platonist perspective would substitute ‘decide’ for ‘define’, thus implicitly suggesting that these concepts can ‘exist’, in the sense of needing to be discovered by some witness-dependent means—eerily akin to a ‘revelation’—even when the domain \(D\) is \(\mathbb{N}\).

We can now inductively assign truth values of ‘satisfaction’, ‘truth’, and ‘falsity’ to the compound formulas of a first-order theory \(S\) under the interpretation \(\mathcal{I}_{S(D)}\) in terms of only the satisfiability of the atomic formulas of \(S\) over \(D\) as follows:\(^{68}\)

**Definition 15** A denumerable sequence \(s\) of \(D\) satisfies \([\neg A]\) under \(\mathcal{I}_{S(D)}\) if, and only if, \(s\) does not satisfy \([A]\);

**Definition 16** A denumerable sequence \(s\) of \(D\) satisfies \([A \rightarrow B]\) under \(\mathcal{I}_{S(D)}\) if, and only if, either it is not the case that \(s\) satisfies \([A]\), or \(s\) satisfies \([B]\);

**Definition 17** A denumerable sequence \(s\) of \(D\) satisfies \([\forall x_i A]\) under \(\mathcal{I}_{S(D)}\) if, and only if, given any denumerable sequence \(t\) of \(D\) which differs from \(s\) in at most the \(i\)’th component, \(t\) satisfies \([A]\);

**Definition 18** A well-formed formula \([A]\) of \(D\) is true under \(\mathcal{I}_{S(D)}\) if, and only if, given any denumerable sequence \(t\) of \(D\), \(t\) satisfies \([A]\);

**Definition 19** A well-formed formula \([A]\) of \(D\) is false under \(\mathcal{I}_{S(D)}\) if, and only if, it is not the case that \(D\) is true under \(\mathcal{I}_{S(D)}\).

It follows that:\(^{69}\)

\(^{67}\)In the sense of [Mu91].

\(^{68}\)Compare [Me64], p.51; [Mu91].

\(^{69}\)cf. [Me64], pp.51-53.
**Theorem 2** (Satisfaction Theorem) If, for any interpretation $I_S(D)$ of a first-order theory $S$, there is a Satisfaction Method $SM(I_S(D))$ which holds for a witness $W_D$ of $D$, then:

(i) The $\Delta_0$ formulas of $S$ are decidable as either true or false over $D$ under $I_S(D)$;

(ii) If the $\Delta_n$ formulas of $S$ are decidable as either true or as false over $D$ under $I_S(D)$, then so are the $\Delta(n+1)$ formulas of $S$.

**Proof** It follows from the above definitions that:

(a) If, for any given atomic formula $[A(x_1, x_2, \ldots, x_n)]$ of $S$, it is decidable by $W_D$ whether or not a given denumerable sequence $(a_1, a_2, \ldots)$ of $D$ satisfies $[A(x_1, x_2, \ldots, x_n)]$ in $D$ under $I_S(D)$ then, for any given compound formula $[A^1(x_1, x_2, \ldots, x_n)]$ of $S$ containing any one of the logical constants $\neg, \rightarrow, \forall$, it is decidable by $W_D$ whether or not $(a_1, a_2, \ldots)$ satisfies $[A^1(x_1, x_2, \ldots, x_n)]$ in $D$ under $I_S(D)$;

(b) If, for any given compound formula $[B^n(x_1, x_2, \ldots, x_n)]$ of $S$ containing $n$ of the logical constants $\neg, \rightarrow, \forall$, it is decidable by $W_D$ whether or not a given denumerable sequence $(a_1, a_2, \ldots)$ of $D$ satisfies $[B^n(x_1, x_2, \ldots, x_n)]$ in $D$ under $I_S(D)$ then, for any given compound formula $[B^{(n+1)}(x_1, x_2, \ldots, x_n)]$ of $S$ containing $n + 1$ of the logical constants $\neg, \rightarrow, \forall$, it is decidable by $W_D$ whether or not $(a_1, a_2, \ldots)$ satisfies $[B^{(n+1)}(x_1, x_2, \ldots, x_n)]$ in $D$ under $I_S(D)$;

We thus have that:

(c) The $\Delta_0$ formulas of $S$ are decidable by $W_D$ as either true or false over $D$ under $I_S(D)$;

(d) If the $\Delta_n$ formulas of $S$ are decidable by $W_D$ as either true or as false over $D$ under $I_S(D)$, then so are the $\Delta(n+1)$ formulas of $S$.

In other words, if the atomic formulas of of $S$ interpret under $I_S(D)$ as decidable with respect to the Satisfaction Method $SM(I_S(D))$ by a witness $W_D$ over some domain $D$, then the propositions of $S$ (i.e., the $\Pi_n$ and $\Sigma_n$ formulas of $S$) also interpret as decidable with respect to $SM(I_S(D))$ by the witness $W_D$ over $D$.

### 7.1 An explicit definition of the standard interpretation of PA

We now consider the application of Tarski’s definitions to the standard interpretation $I_{PA(N, \text{Standard})}$ of PA where:

(a) we define $S$ as PA with standard first-order predicate calculus as the underlying logic;

(b) we define $D$ as $\mathbb{N}$;

(c) we take $SM(I_{PA(N, \text{Standard})})$ as a deterministic Turing machine.

We note that:
Theorem 3 The atomic formulas of PA are algorithmically verifiable under the standard interpretation \( \mathcal{I}_{PA}(N, \text{Standard}) \).

Proof If \([A(x_1, x_2, \ldots, x_n)]\) is an atomic formula of PA then, for any given denumerable sequence of numerals \([b_1, b_2, \ldots]\), the PA formula \([A(b_1, b_2, \ldots, b_n)]\) is an atomic formula of the form \([c = d]\), where \([c]\) and \([d]\) are atomic PA formulas that denote PA numerals. Since \([c]\) and \([d]\) are recursively defined formulas in the language of PA, it follows from a standard result\(^7\) that, if PA is consistent, then \([c = d]\) interprets as the proposition \(c = d\) which either holds or not for a witness \(W\) in \(N\).

Hence, if PA is consistent, then \([A(x_1, x_2, \ldots, x_n)]\) is algorithmically verifiable since, for any given denumerable sequence of numerals \([b_1, b_2, \ldots]\), we can define a deterministic Turing machine \(TM\) that will compute \([A(b_1, b_2, \ldots, b_n)]\) and halt on null input as evidence that the PA formula \([A(b_1, b_2, \ldots, b_n)]\) is decidable under the interpretation.

The theorem follows.

We thus have that:

Corollary 2 The ‘satisfaction’ and ‘truth’ of PA formulas containing logical constants can be defined under the standard interpretation of PA in terms of the evidence provided by the computations of a deterministic Turing machine.

7.2 Defining ‘algorithmic truth’ under the standard interpretation of PA and interpreting the PA axioms

Now we note that, in addition to Theorem 3:

Theorem 4 The atomic formulas of PA are algorithmically computable under the standard interpretation \( \mathcal{I}_{PA}(N, \text{Standard}) \).

Proof If \([A(x_1, x_2, \ldots, x_n)]\) is an atomic formula of PA then we can define a deterministic Turing machine \(TM\) that will compute \([A(x_1, x_2, \ldots, x_n)]\) and halt on any given denumerable sequence of numerals \([b_1, b_2, \ldots]\) as evidence that the PA formula \([A(b_1, b_2, \ldots, b_n)]\) is decidable under the interpretation.

The theorem follows.

This suggests the following definitions:

Definition 20 A well-formed formula \([A]\) of PA is algorithmically true under \( \mathcal{I}_{PA}(N, \text{Standard}) \) if, and only if, there is a deterministic Turing machine which computes \([A]\) and provides evidence that, given any denumerable sequence \(t\) of \(N\), \(t\) satisfies \([A]\);

Definition 21 A well-formed formula \([A]\) of PA is algorithmically false under \( \mathcal{I}_{PA}(N) \) if, and only if, it is not algorithmically true under \( \mathcal{I}_{PA}(N) \).

The significance of defining ‘algorithmic truth’ under \( \mathcal{I}_{PA}(N, \text{Standard}) \) as above is that:

\(^7\)For any natural numbers \(m, n\), if \(m \neq n\), then PA proves \([\neg(m = n)]\) (Me64, p.110, Proposition 3.6). The converse is obviously true.
Lemma 5 The PA axioms PA₁ to PA₈ are algorithmically computable as algorithmically true over \( \mathbb{N} \) under the interpretation \( I_{PA(N, \text{Standard})} \).

Proof Since \([x + y] \), \([x \times y] \), \([x = y] \), \([x'] \) are defined recursively\(^{71}\), the PA axioms PA₁ to PA₈ interpret as recursive relations that do not involve any quantification. The lemma follows straightforwardly from Definitions 14 to 19 in Section 7 and Theorem 3.

Lemma 6 For any given PA formula \([F(x)]\), the Induction axiom schema \([(F(0)) \rightarrow ((\forall x)(F(x) \rightarrow F(x'))) \rightarrow (\forall x)F(x)]\) interprets as algorithmically true under \( I_{PA(N, \text{Standard})} \).

Proof By Definitions 14 to 21:

(a) If \([F(0)]\) interprets as algorithmically false under \( I_{PA(N, \text{Standard})} \) the lemma is proved.

Since \([F(0) \rightarrow ((\forall x)(F(x) \rightarrow F(x'))) \rightarrow (\forall x)F(x)]\) interprets as algorithmically true if, and only if, either \([F(0)]\) interprets as algorithmically false or \([(\forall x)(F(x) \rightarrow F(x'))) \rightarrow (\forall x)F(x)]\) interprets as algorithmically true.

(b) If \([F(0)]\) interprets as algorithmically true and \([(\forall x)(F(x) \rightarrow F(x'))]\) interprets as algorithmically false under \( I_{PA(N, \text{Standard})} \), the lemma is proved.

(c) If \([F(0)]\) and \([(\forall x)(F(x) \rightarrow F(x'))]\) both interpret as algorithmically true under \( I_{PA(N, \text{Standard})} \), then by Definition 20 there is a deterministic Turing machine \( TM \) that, for any natural number \( n \), will give evidence that the formula \([F(n) \rightarrow F(n')]\) is true under \( I_{PA(N, \text{Standard})} \).

Since \([F(0)]\) interprets as algorithmically true under \( I_{PA(N, \text{Standard})} \), it follows that there is a deterministic Turing machine \( TM \) that, for any natural number \( n \), will give evidence that the formula \([F(n)]\) is true under the interpretation.

Hence \([(\forall x)F(x)]\) is algorithmically true under \( I_{PA(N, \text{Standard})} \).

Since the above cases are exhaustive, the lemma follows.

The Poincaré-Hilbert debate: We note that Lemma 6 appears to settle the Poincaré-Hilbert debate\(^ {72}\) in the latter’s favour. Poincaré believed that the Induction Axiom could not be justified finitarily, as any such argument would necessarily need to appeal to infinite induction. Hilbert believed that a finitary proof of the consistency of PA was possible.

Lemma 7 Generalisation preserves algorithmic truth under \( I_{PA(N, \text{Standard})} \).

Proof The two meta-assertions:

\(^{71}\)cf. \( \text{Go31} \), p.17.

\(^{72}\)See \( \text{Hi27} \), p.472; also \( \text{Br13} \), p.59; \( \text{We27} \), p.482; \( \text{Pa71} \), p.502-503.
"[F(x)] interprets as algorithmically true under $I_{PA(N, \text{Standard})}$" and

"[(\forall x)F(x)] interprets as algorithmically true under $I_{PA(N, \text{Standard})}$" both mean:

$[F(x)]$ is algorithmically computable as always true under $I_{PA(N, \text{Standard})}$.

It is also straightforward to see that:

**Lemma 8** Modus Ponens preserves algorithmic truth under $I_{PA(N, \text{Standard})}$.

We thus have that:

**Theorem 5** The axioms of PA are always algorithmically true under the interpretation $I_{PA(N, \text{Standard})}$, and the rules of inference of PA preserve the properties of algorithmic satisfaction/truth under $I_{PA(N, \text{Standard})}$, without appeal to Aristotle’s particularisation.

We conclude that the concepts of Algorithmic verifiability and Algorithmic computability are well-defined under the standard interpretation of PA.

8 Appendix A: Gödel’s Theorem VII(2)

(Excerpted from [Go31] pp.29-31.)

Every relation of the form $x_0 = \phi(x_1, \ldots, x_n)$, where $\phi$ is recursive, is arithmetical and we apply complete induction on the rank of $\phi$. Let $\phi$ have rank $s(s > 1)$. . . .

$\phi(0, x_2, \ldots, x_n) = \psi(x_2, \ldots, x_n)$

$\phi(k + 1, x_2, \ldots, x_n) = \mu[k, \phi(k, x_2, \ldots, x_n), x_2, \ldots, x_n]$

(wher $\psi, \mu$ have lower rank than $s$).

. . . we apply the following procedure: one can express the relation $x_0 = \phi(x_1, \ldots, x_n)$ with the help of the concept “sequence of numbers” ($f$) in the following manner:

$x_0 = \phi(x_1, \ldots, x_n) \sim (\exists f)\{f_0 = \psi(x_2, \ldots, x_n) \& (\forall k)(k < x_1 \rightarrow f_{k+1} = \mu[k, f_k, x_2, \ldots, x_n]) \& x_0 = f_{x_1}\}$

If $S(y, x_2, \ldots, x_n), T(z, x_1, \ldots, x_{n+1})$ are the arithmetical relations which, according to the inductive hypothesis, are equivalent to $y = \psi(x_2, \ldots, x_n)$, and $z = \mu(x_1, \ldots, x_{n+1})$ respectively, then we have:

$x_0 = \phi(x_1, \ldots, x_n) \sim (\exists f)\{S(f_0, x_2, \ldots, x_n) \& (\forall k)[k < x_1 \rightarrow T(f_{k+1}, k, x_2, \ldots, x_n)] \& x_0 = f_{x_1}\}$

(17)

73See Definition 18

74$f$ denotes here a variable whose domain is the sequence of natural numbers. The $(k+1)$st term of a sequence $f$ is designated $f_k$ (and the first, $f_0$).
Now we replace the concept “sequence of numbers” by “pairs of numbers” by correlating with the number pair \( n, d \) the sequence of numbers \( f^{(n,d)}(k) = [n]_{1+(k+1)d}, \) where \([n]_p\) denotes the smallest non-negative remainder of \( n \) modulo \( p \).

Then:

Lemma 1: If \( f \) is an arbitrary sequence of natural numbers and \( k \) is an arbitrary natural number, then there exists a pair of natural numbers \( n, d \) such that \( f^{(n,d)} \) and \( f \) coincide in their first \( k \) terms.

Proof: Let \( l \) be the greatest of the numbers \( k, f_0, f_1, \ldots, f_{k-1} \). Determine \( n \) so that
\[
 n \equiv f_i \mod (1 + (i + 1)!)
\]
for \( i = 0, 1, \ldots, k - 1, \) which is possible, since any two of the numbers \( 1 + (i + 1)! \) are relatively prime. For, a prime dividing two of these numbers must also divide the difference \( (i_1 - i_2)! \) and therefore, since \( i_1 - i_2 < l \), must also divide \( l! \), which is impossible. The number pair \( n, l! \) fulfills our requirement.

Since the relation \( x = [n]_p \) is defined by
\[
 x \equiv n \mod p \& x < p
\]
and is therefore arithmetical, then so also is the relation \( P(x_0, x_1, \ldots, x_n) \) defined as follows:
\[
P(x_0, x_1, \ldots, x_n) \equiv (\exists n, d) \{ S([n]_{d+1}, x_2, \ldots, x_n) \& (\forall k)[k < x_1 \rightarrow T([n]_{1+d(k+2)}, k, [n]_{1+d(k+1)}, x_2, \ldots, x_n]) \& x_0 = [n]_{1+d(x_1+1)} \}
\]
which, according to (17) and Lemma 1, is equivalent to \( x_0 = \phi(x_1, \ldots, x_n) \) (in the sequence \( f \) in (17) only its values up to the \((x+1)\)th term matter). Thus, Theorem VII(2) is proved.

Comment: Gödel’s remark that “in the sequence \( f \) in (17) only its values up to the \((x+1)\)th term matter” is significant. The proof of Theorem VII(2) depends upon the fact that the equivalence between \( f^{(n,d)} \) and \( f \) cannot be extended non-terminatingly.

9 Appendix B: A provability Thesis

The significance of defining classical foundational concepts in a constructive and intuitionistically unobjectionable way, of weakening Church’s Thesis to an equivalence (from an identity), and of supplementing it with an Arithmetical Provability Thesis (that postulates a bridge between formal arithmetical provability and Turing computability of arithmetical formulas), is seen in the following argument.

Thesis 1 Arithmetical Provability Thesis: If the PA formula \( F(x) \) interprets as the relation \( F^*(x) \) in some interpretation \( M \) of a PA, and there is an algorithm that, for any given element \( k \) in \( M \), will determine that \( F(k) \) holds in \( M \), then:

\[
 [F(x)] \text{ is PA-provable.}
\]
The Arithmetical Provability Thesis thus postulates that if a total arithmetical relation $F^*(x)$ 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 $[F(x)]$ in PA.\footnote{We consider elsewhere \cite{An12} the question of whether or not there is a case for validating the Arithmetical Provability Thesis.}

\textbf{Theorem 6} The Arithmetical Provability Thesis implies that it is always possible to determine whether a Turing machine will halt or not when computing any partial recursive function $F$.

\textbf{Proof:} We assume that the partial recursive function $F$ is obtained from the recursive function $G$ by means of the unrestricted $\mu$-operator; in other words, that

$$F(x_1, \ldots, x_n) = \mu y(G(x_1, \ldots, x_n, y) = 0).$$

If $[H(x_1, \ldots, x_n, y)]$ expresses $\neg(G(x_1, \ldots, x_n, y) = 0)$ in PA\footnote{See \cite{Meta2}, p.214}, we consider the PA-provability—and Tarskian-truth\footnote{By definition the interpretation $H^*(x_1, \ldots, x_n, y)$ of $[H(x_1, \ldots, x_n, y)]$ in $M$ is instantiationally equivalent to $\neg(G(x_1, \ldots, x_n, y) = 0)$ \cite{Meta2}, p.117.} in the standard interpretation $M$ of PA—of the arithmetical formula $[H(a_1, \ldots, a_n, y)]$ for a given sequence of numerals $\{[a_1], \ldots, [a_n]\}$ of PA, as below:

(a) Let $Q_1$ be the meta-assertion that the PA-formula $[H(a_1, \ldots, a_n, y)]$ does not interpret as always true in $M$.

Since $G(a_1, \ldots, a_n, y)$ is recursive, it follows that there is some finite $k$ such that any Turing machine $T_1(y)$ that computes $G(a_1, \ldots, a_n, y)$ will halt and return the value 0 for $y = k$.

(b) Let $Q_2$ be the meta-assertion that the PA-formula $[H(a_1, \ldots, a_n, y)]$ interprets as always true in $M$, but there is no Turing machine that computes the corresponding interpreted arithmetical function $H^*(a_1, \ldots, a_n, y)$ as a Boolean function that is always true in $M$.

If we assume the Arithmetical Provability Thesis, then the PA-formula $[H(a_1, \ldots, a_n, y)]$ is unprovable in PA, but always true under interpretation in $M$.

Hence—since $G(a_1, \ldots, a_n, y)$ is recursive—any Turing machine $T_2(y)$ that computes the instantiationally equivalent arithmetical function $H^*(a_1, \ldots, a_n, y)$ as a Boolean function must halt, since its auxiliary tape will return the symbol for self-termination at the first initiation of a non-terminating loop at some $y = k'$.

\textit{Comment:} In his seminal paper on computable numbers\footnote{\cite{Tu36}.} Turing considers the Halting problem, which can be expressed as the query:

\textit{Halting problem for }$T$\footnote{\cite{Me64}, p.256.} Given a Turing machine $T$, can one effectively decide, given any instantaneous description $alpha$, whether or not there is a computation of $T$ beginning with $alpha$?
Turing then shows that the Halting problem is unsolvable by a Turing machine.

Since a function is Turing-computable if, and only if, it is partially Markov-computable, it is essentially unverifiable algorithmically whether, or not, a Turing machine that computes a random, \( n \)-ary, number-theoretic function will halt classically on every \( n \)-ary sequence of natural numbers (for which it is defined) as input, and not go into a non-terminating loop for some natural number input, where:

\[
\text{Non-terminating loop: A non-terminating loop is defined as any repetition of the instantaneous tape description of a Turing machine during a computation.}
\]

and where we note that:

\[
\text{Effective Looping oracle: Any Turing machine } T \text{ can be provided with an auxiliary infinite tape to effectively recognize a non-terminating looping situation; it simply records every instantaneous tape description at the execution of each machine instruction on the auxiliary tape, and compares the current instantaneous tape description with the record. If an instantaneous tape description is repeated, it can be meta-programmed to abort the impending non-terminating loop, and return a meta-symbol indicating self-termination.}
\]

(c) Finally, let \( Q_3 \) be the meta-assertion that the PA-formula \([H(a_1,\ldots,a_n,y)]\) interprets as always true in \( M \), and that any Turing machine that computes the interpreted arithmetical function \( H^*(a_1,\ldots,a_n,y) \) as a Boolean function halts on every natural number input for \( y \), and returns the value \( 0 \) (‘true’).

Now, if we assume an Arithmetical Provability Thesis, then it follows that \([H(a_1,\ldots,a_n,y)]\) is PA-provable.

Let \( h \) be the Gödel-number of \([H(a_1,\ldots,a_n,y)]\). We consider, then, Gödel’s primitive recursive number-theoretic relation \( xBh \) 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 \( T_3(y) \) which computes the characteristic function of \( xBh \), will halt and return the value \( 0 \) (‘true’) for \( x = k'' \).

Since \( Q_1 \), \( Q_2 \) and \( Q_3 \) are mutually exclusive and exhaustive it follows that, when run simultaneously over the sequence 1, 2, 3, \ldots of values for \( y \), one of the parallel trio \((T_1(y) // T_2(y) // T_3(y))\) of Turing machines will always halt for some finite value of \( y \).

---

81 [Me64], p.233, Corollary 5.13 & p.237, Corollary 5.15.
82 An instantaneous tape description describes the condition of the machine and the tape at a given moment. When read from left to right, the tape symbols in the description represent the symbols on the tape at the moment. The internal state \( q_s \) in the description is the internal state of the machine at the moment, and the tape symbol occurring immediately to the right of \( q_s \) in the tape description represents the symbol being scanned by the machine at the moment. ([Me64], p.230, footnote 1).
83 See [?], p.130.
84 It is convenient to visualise the tape of such a Turing machine as that of a two-dimensional virtual-teleprinter, which maintains a copy of every instantaneous tape description in a random-access memory during a computation.
85 [Gos31], p.22, definition 45.
If $T_1(y)$ halts, then a Turing machine will halt when computing the partial recursive function $F$.

If either one of $T_2(y)$ or $T_3(y)$ halts, then a Turing machine will not halt when computing the partial recursive function $F$. □

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

**Corollary 3** The Arithmetical Provability Thesis implies that the parallel trio of Turing machines $(T_1(y) \ // T_2(y) + \text{auxiliary tape} \ // T_3(y))$ is not a Turing machine.

**Corollary 4** The Arithmetical Provability Thesis implies that the classical Church and Turing theses are false.

**References**


Authors postal address: 32 Agarwal House, D Road, Churchgate, Mumbai - 400 020, Maharashtra, India. Email: re@alixcomsi.com, anandb@vsnl.com.