Reviewing current interpretations of classical theory III

A trivial solution to the PvNP problem

Bhupinder Singh Anand

To appear in the proceedings of the 2008 International Conference on Foundations of Computer Science, July 14-17, 2008, Las Vegas, USA.

(*This draft of August 1, 2008 may differ in formatting and other inessential details from the formal paper.*)

Abstract

We show that Gödel has defined an arithmetical relation $R(n)$ which—when treated as a Boolean function—is constructively computable as true for any given natural number n, but which is not Turing-computable as true for any given natural number n. This implies that the current formulation of the PvNP problem admits a trivial logical solution that is not significant computationally.

1 A Provability Theorem for PA

Let $[N]$ denote the usual structure—defined as $\{N$ (*the set of natural numbers*); $=$ (*equality*); $'$ (*the successor function*); $+$ (*the addition function*); $*$ (*the product function*); 0 (*the null element*)$\}$—that serves for a definition of a sound interpretation[1] of first-order Peano Arithmetic, PA.

We first show that PA is 'algorithmically' complete in the sense that:

Provability Theorem for PA: A total arithmetical relation $F(x)$—when treated as a Boolean function—defines a Turing-machine TM_F which computes $F(x)$ as *always* true (i.e., true for any given natural number input n) if, and only if, the corresponding PA-formula $[F(x)]$ is PA-provable.

Proof: It follows from Turing's seminal 1936 paper [Tu36] that every quantifier-free arithmetical function (*or relation, when interpreted as a Boolean function*) $F(x)$ defines a Turing-machine TM_F[2] (cf. [Tu36], pp. 138-139) such that $F(x)$ is TM_F-computable if, and only if, for *any* given natural number n, TM_F will compute $F(n)$ as either true, or as false, over the structure $[N]$ (cf. [An08a]).

[0]Alix Comsi Internet Services Pvt. Ltd. Postal address: 32 Agarwal House, D Road, Churchgate, Mumbai - 400 020, Maharashtra, India. Email: re@alixcomsi.com, anandb@vsnl.com.

[1]An interpretation of PA is sound if all PA-theorems are true under the interpretation. We constructively define such an interpretation of PA in the next section

[2]In the general case, TM_F is defined by the quantifier-free expression in the prenex normal form of $F(x)$.

Now, we can define a finitary interpretation β over $[N]$ (cf. [An08a], [An08c]) where:

(a) $[(\forall x)F(x)]$ is defined as true under the interpretation β if, and only if, the Turing-machine TM_F computes $F(n)$ as *always* true (*i.e., as true for any given natural number* n) under β;
(b) $[(\exists x)F(x)]$ is an abbreviation of $[\neg(\forall x)\neg F(x)]$, and is defined as true under β if, and only if, it is not the case that the Turing-machine TM_F computes $F(n)$ as *always* false (*i.e., as false for any given natural number* n) under β.

β is a sound interpretation of PA over $[N]$ since—when interpreted suitably—all theorems of first-order PA are *constructively*[3] true under β (cf. [An08a]).

Moreover, since β is sound, it defines a finitary model of PA over $[N]$—say M_β—such that:

If $[(\forall x)F(x)]$ is PA-provable, then the Turing-machine TM_F computes $F(n)$ as *always* true (*i.e., as true for any given natural number* n) in M_β;

If $[\neg(\forall x)F(x)]$ is PA-provable, then it is not the case that the Turing-machine TM_F computes $F(n)$ as *always* true (*i.e., as true for any given natural number* n) in M_β.

Moreover, we cannot have that both $[(\forall x)F(x)]$ and $[\neg(\forall x)F(x)]$ are PA-unprovable for some PA-formula $F(x)$, as this would yield the contradiction:

(i) There is a finitary model—say $M1_\beta$—of PA+$[(\forall x)F(x)]$ in which the Turing-machine TM_F computes $F(n)$ as *always* true (*i.e., as true for any given natural number* n).
(ii) There is a finitary model—say $M2_\beta$—of PA+$[\neg(\forall x)F(x)]$ in which it is not the case that the Turing-machine TM_F computes $F(n)$ as *always* true (*i.e., as true for any given natural number* n).

Hence PA is 'algorithmically' complete, in the sense that a total arithmetical relation $F(x)$—when treated as a Boolean function—defines a Turing-machine TM_F which computes $F(x)$ as *always* true if, and only if, the corresponding PA-formula $[F(x)]$ is PA-provable[4][5]. □

1.1 First Tautology Theorem

Now, in his seminal 1931 paper [Go31], Gödel has constructed an arithmetical relation $R(n)$ that is meta-mathematically provable as true for any given natural number n but — since the corresponding PA-formula $[R(x)]$[6] is not PA-provable (cf. [Go31], p25(1))—it follows from the Provability Theorem for PA that there is no Turing-machine TM_R that computes $R(n)$ as *always* true (*i.e. true for any given natural number* n).

[3] In the sense of mechanically, without the use of any human ingenuity.
[4] Note that $[(\forall x)F(x)]$ is PA-provable if, and only if, $[F(x)]$ is PA-provable
[5] Hence PA can have no 'non-standard' model!
[6] Gödel defines, and refers to, this formula by its Gödel-number r (cf. [Go31], p25, eqn.12).

(*By Generalisation*[7]*, stating that the PA-formula* $[R(x)]$ *is not PA-provable is equivalent to stating that the PA-formula* $[(\forall x)R(x)]$[8] *is not PA-provable; the latter is what Gödel actually proved.*)

We thus have the:

First Tautology Theorem: There is no Turing-machine TM_R that computes Gödel's tautology $R(n)$—when treated as a Boolean function—as true for any given natural number n.

1.2 Second Tautology Theorem

However, we also have the:

Second Tautology Theorem: Gödel's tautology $R(n)$ is constructively computable as true for any given natural number n.

Proof: Gödel has defined a primitive recursive relation, $xB_{PA}y$ that holds if, and only if, y is the Gödel-number of a PA-formula, say $[Y]$, and x the Gödel-number of a PA-proof of $[Y]$ ([Go31], p22, dfn. 45).

Since every primitive recursive relation is Turing-computable (*when treated as a Boolean function*), $xB_{PA}y$ defines a Turing-machine TM_B that halts on any natural number values of x and y.

Now, if $g_{[R(1)]}$, $g_{[R(2)]}$, ... are the Gödel-numbers of the PA-formulas $[R(1)]$, $[R(2)]$, ..., it follows that, for any given natural number n, when the natural number value $g_{[R(n)]}$ is input for y, the Turing-machine TM_B must halt for some value of x—which is the Gödel-number of some PA-proof of $[R(n)]$—since Gödel has shown ([Go31], p25(1)) that $[R(n)]$ is PA-provable for any given numeral $[n]$.

Hence $R(n)$ is constructively computable as true for any given natural number n. □

2 P≠NP

Now, one formulation of the PvNP problem is the following: Is there a polynomial time algorithm TM_F that gets as input a Boolean formula F and outputs 1 if and only if F is a tautology?

P≠NP states that there is no such algorithm.

So, if the Gödelian relation $R(n)$ is constructively computable as a tautology, but not recognisable as a tautology by any Turing-machine TM_R, then, by the above reasoning, P≠NP is trivially true logically!

2.1 Is the solution P≠NP significant?

However, the PvNP problem assumes the significance usually accorded to it only to the extent that its solution throws light on the practical consequences that follow from the computational complexity of a number-theoretic relation (*or function*), and not simply from the philosophical consequences of its logical properties.

[7] *Generalisation in PA*: $[(\forall x)A]$ follows from $[A]$.

[8] Gödel defines, and refers to, this formula by its Gödel-number $17Genr$.

The following shows why a trivial logical solution of the PvNP problem — such as that indicated above, which addresses the problem as it is presently formulated — is not significant from a computational complexity perspective.

2.2 Can the PvNP problem be formulated to highlight the significance of computational complexity?

In his paper [Go31], Gödel specifically defined his arithmetical relation $R(n)$ so that it is instantiationally equivalent (cf. [Go31], p29, Theorem VII) to a primitive recursive relation, $Q(n)$, which, of course, is Turing-computable as true by some Turing-machine TM_Q for any given natural number n.

So, even if P≠NP because PA is 'algorithmically' complete—and, therefore, has no non-standard models—we still have two number-theoretic relations that are instantiationally equivalent even though they are not algorithmically identical. The equivalence should, prima facie, suffice to define the computational complexity involved in one case as notionally equal to that actually involved in the other.

2.3 *Why* the PvNP problem may *remain* intractable in ZF

Now, the above distinction between instantiational equivalence and mathematical identity would be absent in any set-theoretic approach to the PvNP problem, since it is an axiom of ZF that two ZF relations (*or functions*) are instantiationally equivalent if, and only if, they are set-theoretically identical.

In other words, if we accept that the consistency of PA implies that P≠NP, then, if ZF is consistent, it can neither reflect that P≠NP nor that P=NP.

This suggests that it may be better to re-formulate the PvNP problem number-theoretically so as to avoid trivial *logical* solutions in PA—or *logical* blind alleys in its interpretation in ZF—and to reflect better its computational significance.

3 Appendix A: Does the 'algorithmic' completeness of PA conflict with Gödel's and Rosser's Theorems?

Of course the conclusion that PA is 'algorithmically' complete seems to conflict with both Gödel's [Go31] and Rosser's [Ro36] interpretations of their non-formal reasoning, each of which separately concludes that PA is 'essentially' incomplete.

However, the conflict resolves itself if we note that:

(i) Gödel's proof of the essential incompleteness of PA ([Go31], Theorem VI, p24) explicitly assumes that PA is ω-consistent;
(ii) Rosser's proof of the essential incompleteness of PA (cf. [Ro36], Theorem II, p233) implicitly presumes that PA is ω-consistent (cf. [An08b]);
(iii) PA is *not* ω-consistent (cf. also [An08c], Appendix A);
(iv) The classical 'standard' interpretation of PA over the structure $[N]$ (cf. [Me64], section §2, pp.49-53; p107) does *not* define a model of PA (cf. [An08c]).

In other words, the apparent conflict does not reflect a logical contradiction, but arises simply from the *non-constructive* - hence intuitionistically objectionable - manner in which Gödel and Rosser, following Hilbert (cf. [An08c]), interpret the truth of the quantified formulas of their formal arithmetic in the classical 'standard' interpretation of PA that is defined over the structure $[N]$.

References

[Cook] Stephen Cook. *The P versus NP Problem*. Official description provided for the Clay Mathematical Institute, Cambridge, Massachusetts.

[Go31] Kurt Gödel. 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.

[Me64] Elliott Mendelson. 1964. *Introduction to Mathematical Logic*. Van Norstrand.

[Ro36] J. Barkley Rosser. 1936. *Extensions of some Theorems of Gödel and Church*. In M. Davis (ed.). 1965. *The Undecidable*. Raven Press, New York. Reprinted from The Journal of Symbolic Logic. Vol.1. pp.87-91.

[Tu36] Alan Turing. 1936. *On computable numbers, with an application to the Entscheidungsproblem*. Proceedings of the London Mathematical Society, ser. 2. vol. 42 (1936-7), pp. 230-265; corrections, Ibid, vol 43 (1937) pp. 544-546. In M. Davis (ed.). 1965. *The Undecidable*. Raven Press, New York. pp. 116-154.

[An08a] Bhupinder Singh Anand. 2008. *A finitary model of PA*. Proceedings of the 2008 International Conference on Foundations of Computer Science, July 14-17, 2008, Las Vegas, USA.

http://alixcomsi.com/19_A_finitary_model_of_PA.pdf

[An08b] Bhupinder Singh Anand. 2008. *Rosser's proof of undecidability implicitly assumes ω-consistency*. (Unpublished draft).

http://alixcomsi.com/11_Rosser_proof_1000.pdf

[An08c] Bhupinder Singh Anand. 2008. *Why Brouwer was justified in his objection to Hilberts unqualified interpretation of quantification*. Proceedings of the 2008 International Conference on Foundations of Computer Science, July 14-17, 2008, Las Vegas, USA.

http://alixcomsi.com/9_Why_Brouwer_was_justified_Rev_1000.pdf