Implicit and explicit examples of the phenomenon of deviant encodings

Paula Quinon
Warsaw University of Technology
Department of Administration and Social Sciences
International Center for Formal Ontology
paula.quinon@pw.edu.pl

Abstract

The core of the problem discussed in this paper is the following: the Church-Turing Thesis states that Turing Machines formally explicate the intuitive concept of computability. The description of Turing Machines requires description of the notation used for the INPUT and for the OUTPUT. Providing a general definition of notations acceptable in the process of computations causes problems. This is because a notation, or an encoding suitable for a computation, has to be computable. Yet, using the concept of computation, in a definition of a notation, which will be further used in a definition of the concept of computation yields an obvious vicious circle. The circularity of this definition causes trouble in distinguishing on the theoretical level, what is an acceptable notation from what is not an acceptable notation, or as it is usually referred to in the literature, "deviant encodings".

Deviant encodings appear *explicitly* in discussions about what is an adequate or correct conceptual analysis of the concept of computation. In this paper, I focus on philosophical examples where the phenomenon appears *implicitly*, in a "disguised" version. In particular, I present its use in the analysis of the concept of natural number. I also point at additional phenomena related to deviant encodings: conceptual fixed points and apparent "computability" of uncomputable functions. In parallel, I develop the idea that Carnapian explications provide a much more adequate framework for understanding the concept of computation, than the classical philosophical analysis.

Keywords: the concept of computation, the concept of natural number,

deviant encoding, computational structuralism, conceptual engineering, explications, the Church-Turing thesis

---

Introduction

The core of the problem discussed in this paper is the following: the Church-Turing Thesis (CTT) states that Turing Machines formally explicate the intuitive concept of computability. The description of Turing Machines requires description of the notation used for the INPUT and for the OUTPUT. The notation used by Turing in the original account and also notations used in contemporary handbooks of computability all belong to the most known, common, widespread notations, such as the standard Arabic notation for natural numbers, the binary encoding of natural numbers or the stroke notation. The choice is arbitrary and left unjustified. In fact, providing such a justification and providing a general definition of notations, which are acceptable for the process of computations, causes problems. This is because a notation or an encoding suitable for a computation, has to be computable. Yet, using the concept of computation in a definition of a notation, which will be further used in a definition of the concept of computation yields an obvious vicious circle.

I use the expression "deviant encoding" to refer to a collection of phenomena related to the impossibility to provide a non-circular account of how to distinguish an acceptable notation (i.e., omega-sequence that is an a priori potential subject of computation) from a non-acceptable notation.

Deviant encodings appear more or less *explicitly* in discussions about what is an adequate or correct conceptual analysis of the concept of computation Shapiro (1982 [23]), Rescorla (2007 [20], 2012 [21]), Copeland & Proudfoot (2010 [3]), Quinon (2014 [16]). Its exact form depends on the underlying picture of mathematics that a given author is working with (realism, nominalism, etc.). Quinon 2018 ([17]) presents an analysis of how three simplified standpoints in philosophy of mathematics deal with the problem of deviant encodings. In **Section 1**, I present an overview of the results from this paper and I also point out to possible further steps that can be taken in the formal analysis of the concept of computation. In **Section 2**, I focus on my main objective in this paper, that is philosophical examples where the phenomenon of deviant encodings appears *implicitly*. I start by analysing those examples where deviant encodings appear in an *implicit* manner in the analysis of the concept of natural number. Thus, I present the position called "computational structuralism". In **Section 3** and **Section 4** I point at additional phenomena related *implicitly* to deviant encodings: conceptual fixed points and apparent "computability" of uncomputable functions. Finally, in the conclusions, I claim that the method of Carnapian explications, introduced already along the lines earlier in the text, provides a much more adequate framework for understanding the concept of computation, than the classical philosophical analysis.

The paper has a rather sketchy character and instead of going into in-depth analysis of one phenomenon, intentionally it only skims the surface of a multiplicity of problems related to inherent circularity of such mathematical concepts as *natural number* or *computation*. My intention is to emphasize points worth further exploring rather than offer trustworthy solutions.

1. Deviant encodings

The expression "deviant encoding" - as used in this paper - covers all sort of limitative phenomena related to the impossibility of providing a fully formal non-circular account of an acceptable notation. Deviations refer to non-computable sequences that cannot be distinguished within the general formal context from sequences that are computable and can be used in computations. In this paper, I use the expression "deviant encoding" independently of the ontological framework within which natural numbers are understood. When the picture gets more fine-grained, deviant encodings can be divided in several categories.

Quinon (2018 [17]) proposes a taxonomy of the "deviation phenomena" that occur while defining the concept of computation. Analyses are conducted for a simplified framework where:

- on the syntactic level there are uninterpreted inscriptions, where functions are string-theoretical generating string values from string arguments;
- on the semantic level there are interpretations that can range from the conceptual content ascribed to initially uninterpreted symbols, to Platonic abstract objects, and where functions are number-theoretical sending numbers to numbers;
- between the two levels there is defined a function of denotation.

Deviations occur on each level. Thus, there exist "deviant encodings" deviations that happen on the syntactic level; "deviant semantics" deviations that happen on the semantic level; "unacceptable denotation function" deviations of the denotation function.

The three-layer picture is inspired by Shapiro (1982 [23]) who searches for an adequate account of what is an "acceptable notation". Shapiro calls "notation" a syntactic sequence of numerals together with a denotation function. This is definitely more handy in *his context*, as he considers only computable sequences of inscriptions. Shapiro's concern is focused on defining adequate ways of associating sequences of numerals with semantical values of natural numbers in such a way that string-theoretic functions have unambiguous number-theoretic counterparts.

Rescorla (2007 [20]) opposes "acceptable notations" in Shapiro's sense, to what he calls "deviant notations", and in this context speaks of denotation functions which associate numerals (symbolic representations of natural numbers) to natural numbers (abstract entities) in a non-computable way. There is a continuum of such mappings. In my terminology proposed above, Rescorla's picture comprises all three types of deviations.

Copeland and Proudfoot (2010 [3]) indicated one of the reasons why deviant encodings might appear in the process of computation, and how using a deviant encoding leads to computing of an uncomputable function. The authors claim that a deviant encoding happens when the omniscient programmer "winks at us" to let us know when the number of Turing Machine (from some standard encoding of Turing Machines), which is being currently processed by some sort of Halting Machine (a machine computing which Turing Machines stop on an input 0), refers to a machine that stops.

> The basic idea of a deviant encoding is easily illustrated. You can give the correct answer to any Yes/No question that I ask you, if it is arranged in advance that I will wink at you (as I ask the question) if and only if the correct answer is Yes. Likewise, a computer is able to answer any and every question if the programmer is permitted to code the answer into the presentation of the question. [3, page 247]

In this way, the Halting Machine computes the halting function, which is an uncomputable function. The "wink" of the omniscient programmer gets encoded in the syntactic structure of the numerals: the numerals representing machines that stop, have a special form, – for instance – are even (their general syntactical form can be reduced to "2n" where "n" is any numeral).

This understanding of what are deviant encodings is clearly different from what I understand by deviant encodings, as the authors mean by a deviant encoding such a standard enumeration of Turing Machines where the encoding is enriched by an extra-formal feature impersonated by the omniscient programmer[1] This is a specific case of a more general problem where deviant encodings refer to encodings representing natural numbers. There obviously are some similarities. The authors themselves observe, while referring to work of Rescorla, their understanding of what are deviant encodings, reminds — not only by name — of the deviant encodings that Rescorla or Shapiro speak about. Certainly, deviant encodings both in the sense of Copeland and Proudfoot, and in the sense of Rescorla, relate to such a notation that enables computing uncomputable functions. The notation is engineered by the omniscient programmer in the case of Copeland and Proudfoot, and is a result of the impossibility to distinguish computable from non-computable sequences, in the case of Rescorla. Also in both cases, deviant encodings lead us towards realist bases: the omniscient programmer magically knows which Turing Machines (under, it seems, a standard encoding) stop in the case of Copeland and Proudfoot; a realist insight into what are natural numbers – under Rescorla reading – is necessary to distinguish deviant semantics from acceptable semantics (which in its turn enables making sure that the denotation function is acceptable, and that, in consequence, the syntactical encoding of natural numbers is acceptable).

There are two natural ways of developing the project of analysing the concept of deviant encodings started in Quinon (2018 [17]). The first way consists in studying the concept of computation as used in specific branches of philosophy of mathematics. For instance, it will be interesting to look closer which kind of solution for the definition of computation is adopted in Platonism, what can be done in constructivism, or whether fictionalism, that assumes that no mathematical objects exist, is also confronted with the problem of deviant encodings. The second way consists in looking at various intensionally different models of computation and analyse in which form the vicious circle appears in them. There are multiple *formal* models

[1]Most possibly the omniscient programmer can be formalised in terms of a Turing oracle.

of computation. All formal models of computation can be proved to be *extensionally* equivalent: they capture the same functions, such as "identity", or "the next element of the sequence". However, models of computation differ *intensionally*: computations on abstract natural numbers are intensionally different from computations performed by a machine using concrete electric signals.

1.1. Deviant encodings and various philosophical standpoints

Quinon (2018 [17]) hypothesizes that the vicious circle persists independently of the philosophical standpoint. The author provides an analysis of following standpoints:

- Purely mechanical/syntactical approaches (nominalism, entwined mathematical concepts);
- Notations have meanings (mild realism);
- Semantics comes first (radical realism, platonic insight).

For instance, in its simplest form the problem presents itself as follows:

> The problem in its purely syntactical version can be formulated as follows. In a definition of Turing computability, one of the aspects that needs to be clarified is the characterization of notation that can be used as an input for a machine to process. If a Turing Machine is supposed to explicate the intuitive concept of computability it is necessary to explain, which sequence of numerals can be used as an input without the use of the concept of computability. That means, we cannot simply say: "sequences that can be used as input are the computable ones" as we have not yet defined what it means "to be computable". (Quinon 2018 [17])

Its more complex occurrence can be found in an interesting case of the Semantical Halting Problem. The Semantical Halting Problem was introduced in the context of deviant encodings in (van Heuveln 2000 [28]). Imagine you have encoded Turing machines with some standard non-deviant encoding, and that you believe that symbols have meanings or interpretations. It can happen that even if your syntax is generated in a recursive manner, your semantics is not following any recursive rules. The Halting Machine that processes encodings of Turing Machines is designed to process information on syntax in an algorithmic manner. If inputed with a given non-standard enumeration of Turing machines, the machine will process those non-computable encodings as it were the standard notation. Again, there is no effective way of defining which semantics are acceptable and which are deviant.

To give an example of a philosophical position outside the strict theoretical context, the phenomenon of deviant encodings concerns as well proponents of concrete computations.

> In our ordinary discourse, we distinguish between physical systems that perform computations, such as computers and calculators, and physical systems that don't, such as rocks. Among computing devices, we distinguish between more and less powerful ones. These distinctions affect our behaviour: if a device is computationally more powerful than another, we pay more money for it. What grounds these distinctions? What is the principled difference, if there is one, between a rock and a calculator, or between a calculator and a computer? Answering these questions is more difficult that it may seem. (Piccinini 2010 [12])[2].

1.2. Deviant encodings and intensional differences in models of computation

The Church-Turing thesis consists of transforming a pre-systematic concept "being intuitively computable" (an *explicandum*) into a precise scientific concept "being TM computable" or "being recursive" (an *explicatum*). As such, it follows the general structure of a Carnapian explication. The method of explication was proposed by Rudolf Carnap, most prominently in (1950 [2]), as a procedure for introducing new concepts to scientific or philosophical language. By the method of explication, Carnap writes,

> we mean the transformation of an inexact, prescientific concept, the *explicandum*, into a new exact concept, the *explicatum*. [2, page 3].

The CTT treated as a Carnapian explication accounts for *intensional* differences between provably *extensionally* equivalent models of computation.

[2]See also Piccinini (2015 [13]).

Depending on the clarification of the concept at hand, various models grasp different aspects of what "to compute" means. For this reason, it would be interesting to see which clarification fosters which way out of the deviation phenomenon.

A preliminary list of possible cases to study is the following: Gödel (193? [8]) prioritized Turing's model as Turing's intention to capture the pure process of a mechanical procedure (Gödel believed in existence of a domain-independent "absolute" concept of idealized computation); Soare (1996 [25]) distinguishes between models based on TM and models based on recursive definitions, and Sieg (1997 [24]) discussed details of recursive model explaining which clarification of the concept of computation Church had in mind while working on theory of recursion. Shagrir (2006 [22]) investigates what can be done by an idealized human who computes by means of effective procedures (*e.g.*, Turing (1936 [27])), Kreisel (1987 [10]) *vs.* what a machine can do (*e.g.*, Gandy (1980 [7])). Shapiro (1982 [23]), Rescorla (2007 [20]), Quinon (2014 [16]) look into computations performed on syntactical numerals vs. semantical abstract natural numbers. Finally, Trakhtenbrot (1988 [26]) distinguishes computations defined for hardware vs. computations defined for software.

2. Deviant encodings and computational structuralism

The phenomenon of deviant encodings is a theoretical result which might seem not having any clear relation to the mathematical or philosophical practice. In this paper, I present an example known from discussions in philosophy of mathematics, and through study of this example, I justify "why would we care about deviant encodings".

The main example which is infected by the problem of deviant encodings, that I am going to consider in the paper is the philosophical position called "computational structuralism". Computational structuralism has been formulated as a consequence of the problem of how to single out the standard model of arithmetic. It aims at reconciling two philosophical/mathematical intuitions about the foundations of arithmetic:

- Natural numbers serve to enumerate and compute.
- Natural numbers are amenable to treatment as abstract entities forming a mathematical structure, in the sense of model theory.

The argument of computational structuralism can be reconstructed in the following way. As I understand it, the main objective of computational structuralism consists in providing the precise account on what is the standard model of arithmetic using minimal philosophical resources and minimal ontological engagement (Quinon & Zdanowski 2007 [15]). In the structuralist tradition, the role of the theory used to single out the standard model of arithmetic is played by $PA2$, however philosophical concerns related to the possibility of quantifying over sets or collections raises doubt when it comes to its conceptual thinness. Additionally, Halbach and Horsten (2005 [9]) list also other reasons for which $PA2$ might be considered problematic. In consequence, computational structuralism opts for using $PA1$ which uses minimal quantificational ressources.

However, there is an important problem related to the use of $PA1$, namely that it has nonstandard models. A nonstandard enumerable model of arithmetic is a model which is not isomorphic to the standard model. It can be bijectively mapped onto the standard structure, it falls under the axiomatic description, but the order on the set of its elements differs essentially from the order on the standard model. The order on the standard model is called an ω-order, that is, it corresponds to the order of the natural numbers progression. I will call, in a usual way, N, the order on the natural number progression, Z, the order of negative and positive integers, and Q, the order of rational numbers. The order on a countable non-standard model starts with elements ordered in N, then it is followed by a dense order of copies of integers, $Q \times Z$. Computational structuralism searches for a way of overcoming the difficulty caused by existence of non-standard models by adding a meta-mathematical constraint about the computability of interpretation of functional symbols in the language, and then it uses Tennenbaum's theorem in order to single out the standard model of arithmetic.

Theorem 2.1 (Tennenbaum 1959). *Let* $\mathcal{M} = \langle \mathbb{M}, +, \times, 0, 1, < \rangle$ *be a enumerable model of* PA1*, and not isomorphic with the standard model* $\mathcal{N} = \langle \mathbb{N}, +, \times, 0, 1, < \rangle$*. Then* $\mathcal{M}$ *is not recursive.*

The contrapositive of this theorem makes its relevance more explicit:

Theorem 2.2 (Tennenbaum transposition). *Let M be an enumerable model of first-order Peano arithmetic. If the interpretation of addition and multiplication within M are computable then M is a standard model for arithmetic (a model with ω–type ordering).*

One of the philosophically interesting consequence of the application of Tennenbaum's theorem is that the set of models singled out with its help are ω-models where ω is computable (Quinon & Zdanowski 2007 [15]). Those models are called "intended" and form a proper subset of standard models.

The vicious circle faced by computational structuralism differs from the vicious circles that are the focus of Quinon (2018 [17]). There, I was only concerned by the concept of natural number being indirectly involved in the definition of what "to compute" means. Conceptual structuralism needs to handle a slightly more elaborate idea. Its objective is to explicate the concept of natural number, identified with the standard model of arithmetic. Its solution consists in using the idea that natural numbers, and in particular those which are defined by Peano's axioms, are the entities used for counting and computing. In consequence, natural numbers are defined in terms of computations. However, this is where the vicious circle arises: one of the characteristic features of the concept of computation is that computation is *always* defined on some given domain.[3] This domain is always identifiable with the structure of natural numbers.

Quinon (2018 [17]) argues that independently of the philosophical standpoint, each solution leads to another vicious circle. In the case of the concept of computation, nesting vicious circles are unavoidable. Researchers working in the area of computational structuralism, where the vicious circle is augmented by the concept of natural number, are more optimistic. Alternative ways out have been proposed. In this paper, I want to focus on the two the most involved in what can be called "conceptual engineering".

The first approach, proposed in, for instance (Quinon & Zdanowski 2007 [15]), proposes to take as basic the concept of computation defined as symbols manipulation. This way of thinking qualifies as an example of application of the method of Carnapian explication. Carnapian explication is used to introduce new concepts to the language in some formal context. It consists in transforming an intuitive concept into a formal concept shaped for a specific scientific context.[4] The intuitive concept of computation as manipulation of symbols, where symbols do not need to form any specific structure, is

[3] As mentioned above, non-realised Gödel's objective consisted in finding an " absolute" concept of computation, *i.e.*, such a concept of computation that does not depend on any domain.

[4] For a discussion of different explications of concepts of computation, see Quinon (2019 [18]).

formalised by the concept of Turing computation. Computation over strings of characters is a primitive notion that does not presuppose either any other notion of computation, or any independent conception of natural numbers.

I do not want to decide here if this argument has or does not have any weak points, and if it is or does not end up in another vicious circle. The method of explication provides a smooth way of providing a solution that is conceptually “good enough”. There can be other ways of explicating the same intuitive concept.[5]

The second approach, comes from (Button & Smith 2012 [1]) and (Dean 2014 [5]). Button and Smith claim that Tennenbaum’s theorem is of no use for a philosopher who wants to distinguish the standard model from other possible models of arithmetic. As the authors say:

> Suffice it to note that our discussion of Tennenbaum’s Theorem illustrates a familiar moral: philosophical problems which are supposedly generated by mathematical results can rarely be tackled by offering more mathematics. [1, page 120]

Their argument is based on the nesting vicious circles phenomenon. They observe, that when the concept “natural number” is explicated, the concepts used in this explication, such as “to compute” or “finite” need explications in their turn, *etc.* Dean (2014 [5]) is similarly sceptical when it comes to the purposefulness of using Tennenbaum’s theorem to single out the standard model of arithmetic. However, differently to Button and Smith, Dean develops a full fledged philosophical position. It is a Putnam-style model-theoretic realism for the concept of computation (see Putnam 1980 [14]). Dean claims that there is no point in trying to find external arguments to distinguish between various standard and non-standard models of arithmetic, nor any recursive theory. We should rather use the richness of the model-theoretic universe for studying structural properties of the concept of computation. Dean claims that it rather shows that there exists a continuum of pairs: model of arithmetic and computation in this model of arithmetic. In consequence, the Tennenbaum’s result instead of contributing to singling out the standard model of arithmetic, indicates that there exist non-computable

[5]Similar way of thinking is suggested by Halbach and Horsten (2005 [9]), however those authors rather attract philosophical attention to the Theorem and they describe several ways of using it depending on the adapted philosophical standpoint.

omega-models of arithmetic (the so called deviant or weird permutations) with a corresponding concept of computation defined within the model.

In the two final sections of this paper, I am going to discuss two additional philosophical phenomena that are related to the solutions proposed to think about the Tennenbaum's result and as such related to deviant encodings. The first is the phenomenon of "conceptual fixed points", the second is the proof that each arithmetical function is provably computable in some model of arithmetic.

3. Natural numbers, computation, and conceptual fixed points

In moral philosophy, "the moral fixed points" are those moral propositions that are moral truths that need to be incorporated in a moral system. A normative system which fails to incorporate such propositions is not a moral system, but a normative system of some other kind. A useful example of such a moral fixed point is the proposition: "It is wrong to engage in the recreational slaughter of a fellow person" (Cueno & Shafer-Landau 2014 [4]).

Eklund (*e.g.*, 2015 [6, chapter 5]) extends this phenomenon to frameworks outside moral philosophy and, as he calls it, the "thinnest" normative words like "good", "right", "ought". Eklund observes that in each conceptual framework, there exist concepts that are difficult, if not impossible to engineer. "Truth" is one of those concepts. People care about truth, writes Eklund, and they do not care about some conceptually engineered concept of "truth*". In consequence, truth is a concept that should keep a fixed position in a conceptual framework, and refer to the natural kin of assertions and beliefs. Similarly, "existence" is a conceptual fixed point. Eklund rejects the claim present in the contemporary metaontological debate, where it is assumed "that there are alternative notions of existence that can be employed". He claims that, similarly as in the case of "truth", a conceptual framework that would result from adapting a conceptually engineered concept of "existence" would need to adjust its other key concepts in such a way that the resulting framework would be isomorphic to the initial one. Thus, "One cannot, so to speak, *selectively* engineer the quantifier".

Let me now observe the following relation between the conceptual fixed points and fixed points traditionally analysed in mathematics in the context of diagonalisation or self-reference: the conceptual fixed points as defined by Eklund, are the concepts interpreted in, what we call in philosophy of mathematics, their intended models. In different words, a fixed point consists of a pair: *the engineered concept* corresponding to the intended meaning of the concept, or to borrow Eklund's expression – the interpretation that "people care about", and *the possible world of interpretation*, which actually corresponds to the intended model of this concept. Both concepts of natural number and the concept of computation are in this sense.

4. Computing non-computable functions

Dean's solution and the context of model theoretical realism developed by Putnam brings us to another philosophical phenomenon related to the concept of computation. If there is no distinguished or intended model of arithmetic, as Dean suggests, there is no distinguished or intended concept of computation and in consequence, no encoding is deviant. Each encoding corresponds to some model of arithmetic.

The explanation proposed by Dean is related to the meta-arithmetic proofs which demonstrate that it is possible to compute non-computable functions or that it is possible to prove consistency of arithmetic, that makes it to the surface on the regular basis. Most recently, Hamkins has used the diagonal argument to prove that there is a function, which goes through all the non-standard models and computes all functions computable in any of them. Such a proof has been recently published on Hamkins' blog[6].

Hamkins "proves" the following theorem:

Theorem 4.1. *There is a Turing machine T with the following property, for any function $f : \mathbb{N} \mapsto \mathbb{N}$ there is a model of $PA1$ such that in this model, if we give T any standard natural number, it halts and computes $f(n)$.*

In different words, the theorem states that there exists a non-standard model in which the function is computed. Since it is a non-standard model, a computation forces a nonstandard number of steps. I will not be getting into formal details of the proof. My objective is to attract attention to philosophically interesting relations between concepts.

[6]Joel David Hamkins, Mathematics and Philosophy of the Infinite, "Every function can be computable" March 2016, http://jdh.hamkins.org/every-function-can-be-computable/?fbclid=IwAR0kuIR5V2d6PyxTwYEjfgKkRE0ZAKB9L9QleKyV3R5vasPVl76RIauSaOY.

Conclusions

To sum up, this paper is about the close relation between the concept of natural number and the concept of computation. It explores the idea, facilitated by stepping away from conceptual analysis and adapting the method of Carnapian explication, that there exists an intended model of arithmetic. Unlike Dean (2014 [5]), but to some extent following Button and Smith (2012 [1]), I claim that in order to find a pragmatic and executable way out of the vicious circle, one needs to accept that language develops from informal to formal and that concepts are first incomplete, and then those concepts mature[7]. If, as in (Quinon & Zdanowski 2007 [15]) one accepts that there exists a workable and reasonable, conceptually rich, concept of computation understood as symbol manipulation, then the explication is available through the application of Church-Turing thesis and yields TM-computation as its result.[8].

Acknowledgments

This paper belongs to a series of papers devoted to the phenomenon of deviant encodings that have got initiated with the publication of the lecture “Taxonomy of Deviant Encodings” that I had the honour to give on the occasion of Martin Davis’ 90th birthday at the Computability in Europe conference in Kiel in 2018. The series to this date consists of “Taxonomy of Deviant Encodings” [17], this paper, and the paper “The Anti-Mechanism Argument Based on Gödel’s Incompleteness Theorems, Indescribability of the Concept of Natural Number and Deviant Encodings” [19], which is currently under revisions for publication. I would like to thank all those who helped me better understanding the phenomenon of deviant encodings. My acknowledgements start with Stewart Shapiro, Carl Posy, Oron Shagrir, Diane Proutfoot and Jack Copeland, and other participants of the working group “Computability” from Israel Institute for Advanced Studies. I am indebted to Liesbeth de Mol and Giuseppe Primiero for trusting me with the

[7] Similar idea is being developed in the context of open texture, see the account of philosophy of Friedrich Waismann in Makovec & Shapiro (2019 [11]).

[8] Quinon (2019 [18]) explains how to think of the CTT in terms of Carnapian explication; the idea to use the CTT to explain the concept of acceptable notation is used by Shapiro (1982 [23]); TM-computation is provably equivalent to recursivity, but also, it is possible to prove Tennenbaum’s theorem for TM-computations.

presentation of my ideas at CiE. I am indebted to Pawel Stacewicz (WUT), the editor of this special issue, for kindly inviting me to participate in his project. I am infinitely grateful to Patrick Blackburn (Roskilde) who spent with me uncountable hours discussing problems surrounding deviations and philosophical significance of Tennenbaum's theorem. I am also very grateful to Aleksandra Cieslak, who not only always proofreads my texts making sure that all the thoughts are clear, but also asks very relevant philosophical and mathematical questions. I am finally indebted to all the anonymous reviewers of the "deviant" papers for valuable insights.

References

[1] Button, T. & Smith, P. (2012). The Philosophical Significance of Tennenbaum's Theorem. Philosophia Mathematica 20(1): 114–121.

[2] Carnap, R. (1950). Logical Foundations of Probability. Routledge and Kegan Paul.

[3] Copeland, J. & Proudfoot, D. (2010). Deviant encodings and Turing's analysis of computability. Studies in History and Philosophy of Science 41: 247–252.

[4] Cuneo T. & Shafer-Landau R. (2014). The moral fixed points: new directions for moral nonnaturalism. Philosophical Studies 171: 399-443.

[5] Dean, W. (2014). Models and Computability. Philosophia Mathematica 22 (2): 143–166.

[6] Eklund, M. (2015). Intuitions, Conceptual Engineering, and Conceptual Fixed Points. In: Daly C. (ed.). The Palgrave Handbook of Philosophical Methods. Palgrave Macmillan: 363-385.

[7] Gandy, R. (1980). Church's Thesis and Principles for Mechanisms. In: Barwise, J., Keisler H. J. & Kunen, K. (eds.). The Kleene Symposium. North-Holland Publishing Company: 123-148.

[8] Gödel, K. (193?). Undecidable Diophantine Propositions. In: Feferman S., et al. (eds.). Gödel, K., Collected Works, Volume III, Unpublished essays and lectures. Oxford University Press 1995: 164–175.

[9] Halbach, V. & Horsten, L. (2005). Computational Structuralism. Philosophia Mathematica 13(2): 174–186.

[10] Kreisel, G.(1987). Church's thesis and the ideal of informal rigour. Notre Dame Journal of Formal Logic 28: 499–519.

[11] Makovec, D. & Shapiro S. (eds.) (2019). Friedrich Waismann. The Open Texture of Analytic Philosophy. Springer.

[12] Piccinini, G. (2010/2017). Computation in Physical Systems. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Metaphysics Research Lab, Stanford University.

[13] Piccinini, G. (2015). Physical Computation: A Mechanistic Account. Oxford UP.

[14] Putnam, H. (1980). Models and Reality. Journal of Symbolic Logic 45(3): 464–482.

[15] Quinon, P. & Zdanowski, K. (2007). Intended Model of Arithmetic. Argument from Tennenbaum's Theorem. In: Cooper, S. B., Kent, T. F., Löwe, B. & Sorbi, A. (eds.). Computation and Logic in the Real World. CiE: 313–317.

[16] Quinon, P. (2014). From Computability over Strings of Characters to Natural Numbers. In: Olszewski, A.; Brożek, B. & Urbańczyk, P. (eds.). Church's Thesis, Logic, Mind & Nature. Copernicus Center Press: 310–330.

[17] Quinon, P. (2018). Taxonomy of Deviant Encodings, Lecture Notes in Computer Sciences 10963: 338–348.

[18] Quinon, P. (2019). Can Church's Thesis be Viewed as a Carnapian Explication? Synthese: Online First.

[19] Quinon, P. (2020). The Anti-Mechanism Argument Based on Gödel's Incompleteness Theorems, Indescribability of the Concept of Natural Number and Deviant Encodings. Under revisions for publication.

[20] Rescrola, M. (2007). Church's thesis and the conceptual analysis of computability. Notre Dame Journal of Formal Logic 48 (2): 253–280.

[21] Rescrola, M. (2012). Copeland and Proudfoot on computability. Studies in History and Philosophy of Science Part A 43 (1): 199–202.

[22] Shagrir, O. (2006). Gödel on Turing on Computability. In: Olszewski, Adam; Woleński, Jan & Janusz, Robert (eds.), Church's Thesis after 70 years. Ontos-Verlag: 393–419.

[23] Shapiro, S. (1982). Acceptable Notation. Notre Dame Journal of Formal Logic 23(1): 14–20.

[24] Sieg, W. (1997). Step by Recursive Step: Church's Analysis of Effective Calculability. The Bulletin of Symbolic Logic 3 (2): 154-180.

[25] Soare, R. (1996). Computability and Recursion. Bulletin of Symbolic Logic 2: 284—321.

[26] Trakhtenbrot, B. (1988). Comparing the Church and Turing approaches: two prophetical messages. In: Herken, R. (ed.) The universal Turing machine: a half-century survey. Oxford University Press: 603-630.

[27] Turing, A. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society 42: 230–265; correction in (1937) 43: 544–546; reprinted in (Davis 1965: 115–154); page numbers refer to the (1965) edition.

[28] van Heuveln, B. (2000). Emergence and consciousness: Explorations into the Philosophy of Mind via the Philosophy of Computation. Ph.D. thesis. State University of New York at Binghampton.