A Taxonomy of Deviant Encodings*

Paula Quinon[1] [0000−0001−7574−6227]

Philosophy Department, Lund University, Lund, Sweden
paula.quinon@fil.lu.se
http://www.paulaquinon.com

Abstract. The main objective of this paper is to design a common background for various philosophical discussions about adequate conceptual analysis of "computation".

Keywords: the Church-Turing thesis · deviant encodings · fixed points of conceptual analysis.

1 Introduction: Circularity in the definition of computability

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`. 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 standard Arabic notation for natural numbers, binary encoding of natural numbers or 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 so, because the comprehensive definition states that such a notation or encoding has to be computable. Yet, using the concept of computability in a definition of a notation, which will be further used in a definition of the concept of computability yields an obvious vicious circle.

This argument appears in discussions about what is an adequate or correct conceptual analysis of the concept of computability. Its exact form depends on the underlying picture of mathematics that an author is working with. My objective in this paper is, firstly to discuss various versions and aspects of this problem, and then, secondly, to point towards possible solutions, both those

* I am indebted to Liesbeth de Mol and Giuseppe Primiero for inviting me to the Special Session in History and Philosophy of Computing at CiE 2018. I am also grateful to Patrick Blackburn (Roskilde) and Nina Gierasimczuk (KTH) for helpful comments on an earlier version of this paper. I am finally indebted to the anonymous reviewer for the valuable insight into possible new openings to which the subject matter of the paper can, and certainly will, lead.

proposed explicitly, and these which are relevant for us only implicitly, because they target some disguised version of the problem. Finally, I will indicate related topics and formulate some ideas arising from this discussion.

2 Background: Philosophical framework and terminology clarification

It was Gödel's ambition to explicate the concept of "absolute" computation, that is a computation which is not formulated for a particular domain. However, all know today explications of computability refer to some sort of ω-progression. Philosophical investigations into the concept of natural number, and its use as basis for analysis of the concept of computation, involve – more or less explicitly – one or both layers:

1. the *syntactical* layer of numerals and
2. the *semantical* layer of abstract natural numbers.

A *denotation* δ maps the elements from one layer to the elements of the other layer. See [16].

The syntactical layer of numerals consists of a sequence of inscriptions susceptible to be subjected to computational manipulations. Various syntactical systems have different syntactical properties. *A priori*, without an interpretation, a sequence of random inscriptions is as good as a binary notation based on recursive combinations of zeros and ones. At the semantical layer natural numbers are always understood as some sort of abstract entities. It does not belong to the current endeavor to analyze all possible modalities of existence of abstract natural numbers, neither to decide, which of these modalities is the correct one. That would complicate the picture unnecessarily. However, in order to mark the ground, let me say that you can think of variety of things starting from types of inscriptions, conceptual content associated with numerals in the process of learning, cognitive content issued from the empirical experience, a system of mental representations or Platonic ideas remaining in the Platonic world of ideas. The denotation function matches numerals from the syntactical layer with natural numbers from the semantical one, so denotation is a cross-modal function, mapping physical inscriptions on abstract objects.

Computations can be defined on the syntactical layer, or semantical layer or on both of them combined. Functions from syntactical layer are called string-theoretic functions, functions from semantical layer are called number-theoretic functions.

This picture was explicitly proposed by [16] and, in a more or less implicit manner, appears in various other papers studying philosophical correlations between computability and natural numbers of such authors as [1], [1996] or [15]. The terminology that various authors use differ from one account to another. My very first objective in this paper is to clarify the basic vocabulary.

Three notions come back in accounts of deviations related to the concept of computability are the following: "notation", "encoding" and "semantics".

For instance, when it comes to "notation", in his paper entitled "Acceptable notation" [16] calls "notation" a syntactic sequence of numerals *together* with a denotational function. This is definitely more handy in his context, as he considers only computable sequences of inscriptions. Shapiro's concern consists in defining adequate ways of associating those sequences with semantics in such a way that string-theoretic functions have clear number-theoretic counterparts. [4] by "notation" mean the triple: a sequence of inscriptions with a denotation function and the standard semantics. One notation differs from another one by syntactical features of the sequence of inscriptions. In this paper, I speak of notation in the context of purely syntactical layer of inscriptions. I speak of notation with denotation function or notation with interpretations when I introduce other layers.

When the full picture (all layers) is in place, I call "deviant encodings" deviations that happens on the syntactic level; I call "deviant semantics" deviations that happens on the semantic level; finally I call "unacceptable denotation function" deviations of the denotation function.

I divide the discussion in this paper to three general perspectives:

- I make an attempt to talk about numerals without any reference to natural numbers formulating a purely syntactical version of the problem;
- I reflect on what happens when one starts associating names with abstract objects, and in this context I look at the denotation function;
- I look at those realistic positions where abstract objects are taken as existing before being named.

3 The lightest shade of the problem: Purely syntactical approach

The general presentation of the problem, proposed at the beginning of this paper, addresses the syntactical version of the issue. In this version, computations are understood as manipulations of inscriptions, functions on inscriptions are string-theoretic functions, *etc.* This version is easily conceptualised in the context of a machine. When a human agent is involved at any stage the computational process, there exists at least a theoretical possibility that purely mechanical computations exist.[1]

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 means "to be computable".

[1] It might be claimed that humans always associate some meaning with symbols.

Study of symbol manipulation as a mathematical endeavor has its place in history of philosophy. A clear distinction between syntax and semantics dates from [17]. In a different context, [6] designs methods to deal with a purely syntactical calculus. [10] introduced philosophical problems related to the rejection of all abstract entities.

To understand this situation better, ask this question: what one could tell the Skeptic, who thinks that there is no way of distinguishing an acceptable notation from an unacceptable one?

3.1 Nominalist Platonism solution: Syntactic entwining

One could tell the Skeptic that sequences of inscriptions exist and are free of interpretations independently of human beings apprehending them. However, even if we cannot see that from our earthy perspective, there is something specific about certain sequences of inscriptions. Some of the sequences are computable. Hence, this primarly nominalistic position has a Platonist dimension. I call it "Platonist Nominalism"[2].

Imagine that, similarly to the librarians from the Library of Babel, described by Borges - I am going to get back to the original Borges story a little bit later - you wander in the land of inscriptions. You might be a robot equipped with means to process sequences of inscriptions and after processing some of those sequences you get an expected `output`. You can also imagine a "Chinese Room world"[3]. In any of those cases, you have no means to formulate a general theoretical law enabling you to distinguish processable sequences from non-processable ones.

"Nominalist Platonism" faces similar problems as those encountered by concrete computations. Consider, for instance, presentation of concrete computations by [13].

> 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.

The difficulty for both, nominalist Platonism and concrete computations, consists in distinguishing sequences of physical objects (inscriptions in the first

[2] The name "nominalist Platonism" has been used in a different context by George Boolos in "Nominalist Platonism", *Philosophical Review* 94 (3):327-344 (1985). I do not want to get into comparison here.

[3] John Searle (1980), "Minds, Brains and Programs", Behavioral and Brain Sciences, 3 (3): 417457.

case and whatever physical systems, in the second) that can play the role of subjects of computations from those that cannot. We know the difference between a rock and a computer, in as much the same way as we know the difference between computable and non-computable sequences of inscriptions. We make a difference between notations for natural numbers, random sequences of random inscriptions, and even weird permutations of numerals.

However, when it comes to formulating an explanation of why is this so, there is no easy way out. Nominalist Platonism suffers from a severe and incurable epistemological problem of the sort described in [2] in the context of full-blooded Platonism: we always face a choice between realistic ontology and epistemological access to abstract objects. It is not impossible to get both. "[S]omething must be said to bridge the chasm, created by [...] [a] realistic [...] interpretation of mathematical propositions... and the human knower, he writes. For prima facie "the connection between the truth conditions for the statements of [our mathematical theories] and [...] the people who are supposed to have mathematical knowledge cannot be made out." [2, page 675], see also [7].

3.2 Analytic solution: The Turing Blank-Type Restriction and Turing Notational Thesis

Some Skeptics will obviously not agree to accept the Nominalist Platonism solution and we shall not take it against them. The position is plausible and coherent, but it is not philosophically very fruitful. The lack of epistemological access is a strong and non fixable defect rendering Nominalist Platonism useless for everyday reasonings, decision making and, most importantly, computational practice.

In this paper, I will recall a solution that has been proposed in the context of computability. The answer that might help overcome Skeptics' worries was originally proposed by Turing and recently reevaluated by Copeland & Proudfoot [8]. I will call it the "analytic solution".

Copeland & Proudfoot reconstruct the way in which Turing implicitly guarantees that sequences of symbols that are processed by a machine are computable. Computable sequences are subjects of two constraints.

Firstly, the *Turing's Blank-Tape Restriction*: "If the [Turing] machine is supplied with a blank tape and set in motion [...] the subsequence of the symbols printed by it which are of the first kind [i.e. binary digits] will be called the sequence computed by the machine. The real number whose expression as a binary decimal is obtained by prefacing this sequence by a decimal point is called the number computed by the machine" [18, page 232].

Secondly, *Turing Notational Thesis*: "Any job of work that can be done by a human computer engaged in numerical calculation can be carried out equivalently by a human computer employing Turing-unary notation."

When these two constraints are put together, the sequence that is in the input and the sequence that is in the output of the process are necessarily computable.

4 The problem gets one shade darker: Numerals get meanings

Let's now imagine that the Skeptic is convinced by the arguments that one can generate the sequence of inscriptions, - or, more precisely, of numerals - in such a way that this structure is recursive. She can do it on a basis that she finds the most convincing (social, cognitive, Blank-Type Restriction, *etc.*), it will not play any role in my further reflections.

Now, let us assume that the Skeptic thinks that numerals have meanings, but she refuses to rely on arbitrary choices of a "standard" notation, a "standard" denotation function, and a "standard" semantics. Instead, she asks for providing her a way to distinguish acceptable notations from unacceptable ones. In consequence, the problem of deviation extends to the semantical level.

4.1 Everyday solution: Arbitrary choice of notation and denotation

The solution commonly adopted in the real world is to take arbitrary decisions, possibly based on social, cultural or cognitive reasons, of which notation and which denotation to use. We simply use a commonly known, well-recognised notation, whose denotation does not leave to us any doubts. This is the case of [4]. When those authors introduce the question of changing notation they narrow the problem to the practical issue of using rather one notation than another. For instance, they say, "multiplying numbers given in decimal numerals (expressing the product in the same form) is easier in practice than multiplying numbers given in something like Roman numerals". They also claim that "it is possible in principle to do it in any other notation, simply by translating the data from the difficult notation into an easier one, performing the operation using the easier notation, and then translating the result back from the easier to the difficult notation", which means there is a computational translation between the notations. Ideas of "deciphering" a notation and "the system now in common use" show that, according to the authors, defining a notation require an external, non-mathematical, non-formal, non-effective knowledge [4, page 24].

Indeed, when they define a Turing machine they say:

> A Turing machine is a specific kind of idealised machine for carrying out computations, especially computations on positive integers represented in monadic notation.

There is no theoretical explanation that would justify the choice of one notation together with a denotation function over another. One can, obviously, try to justify the choice by saying that numerals are social creations that children learn in social interactions or that there is a necessity of learning them in this specific order because of cognitive process, but all these justifications go beyond the theoretical and conceptual context, which is ours.

4.2 Formal problem: Deviant encodings and deviant semantics

It is highly doubtful if any Skeptic gets convinced by an arbitrary argument. There is no theoretical reason to favour one notation, accompanied by a denotation, over another. In consequence, the "lacuna" remains, but at another level. And here finally comes time to recall the story of the Library of Babel. Under Jorge Luis Borges description [1941] "the Library is total and that its shelves register all the possible combinations of the twenty-odd orthographical symbols (a number which, though extremely vast, is not infinite). [...] When it was proclaimed that the Library contained all books, the first impression was one of extravagant happiness. All men felt themselves to be the masters of an intact and secret treasure. [...] As was natural, this inordinate hope was followed by an excessive depression. The certitude that some shelf in some hexagon held precious books and that these precious books were inaccessible, seemed almost intolerable."

The librarians usually know which books make sense and which do not. It is ok if a copy contains typos. However, the librarians do not understand reasons why some sequences are special. *A priori*, no one is able to tell the difference between books that tell the stories from books containing illegible information. Similarly, no one is able to tell the difference between sequences that are computable from those that are not.

However, if no external justification is adapted, we fall once more into vicious circle of a conceptual analysis. This type of deviations is a deviation of the meaning associated to the denotation function. There is no formal way of defining such a function without referring to the concept of computability. A traditional way of presenting this problem is clearly visible on the example of the Semantical Halting Problem introduced by [12] and discussed by [8].

The classical formulation of the Halting Problem, first described by [18] and named by Davis [1958, pages 70–71], provides a negative reply to the question of whether there exists a general procedure to decide if a given Turing Machine, or more generally a given computer program, will eventually stop. The proof goes by showing that assuming the opposite leads to contradiction. The classical formulation uses some intended and arbitrary, recursive, notation with standard semantics, *e.g.,* Arabic notation with standard interpretations.

Formulating the Halting Problem in purely syntactical terms is not really possible, because the input, even if purely syntactical, is being generated by a recursive procedure of encoding Turing Machines. Each numeral stands for the Turing Machines it encodes. Problems arise however, when semantical layer gets involved in the Semantical Halting Problem.

The Semantical Halting Problem is a Skeptical issue of the same sort as the problem of non-computable, but ω-ordered models of the first-order Peano Arithmetic introduced by [11] and [14], and discussed by [5]. In the presentation of the classical Halting Problem, machines are encoded in a standard way. The Semantical Halting Problem opens up to the possibility of using deviant - non-computable - encodings. Imagine, you have encoded Turing machines with some standard encoding. Then, it is Funes de Memorious – a character from another

Borges' story – who inherits the job from you. He is not following any recursive rules. Since his memory is infinite and he has trouble synthesising information, he names subsequent machines by a random name, and symbolises by a random inscription.

> He told me that in 1886 he had invented an original system of numbering and that in a very few days he had gone beyond the twenty-four-thousand mark. [...] In place of seven thousand thirteen, he would say (for example) *Máximo Pérez*, in place of seven thousand fourteen, *The Railroad*; other numbers were *Luis Melián Lafinur*, *Olimar*, *sulphur*, *the reins*, *the whale*, *the gas*, *the caldron*, *Napoleon*, *Augustn de Veida*. In place of five hundred, he would say *nine*. Each word had a particular sign, a kind of mark; the last in the series were very complicated... I tried to explain to him that this rhapsody of incoherent terms was precisely the opposite of a system of numbers. I told him that saying 365 meant saying three hundreds, six tens, five ones, an analysis which is not found in the "numbers" *The Negro Timoteo* or *meat blanket*. Funes did not understand me or refused to understand me.

Tapes with Funes encoding are subsequently given to the Halting Machine. What then happens? The Halting Machine that processes encodings of Turing Machines is designed to process information 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 standard notation.

No one will, obviously, have the idea of encoding Turing Machines with a non-standard encoding. However, the problem of distinguishing one encoding from another is the same as it was in the case of purely syntactical version of the problem. If "being computable by a Turing machine" is how computable is defined, one cannot use the concept of being computable in the definition. Is there any formal general way of distinguishing standard encodings from deviant encodings?

4.3 Way out: Constraints on denotation function (Shapiro)

[16] defines computability on inscriptions and then searches for ways of constraining the denotation function in such a way that no uncomputable semantics can be reached. The first constraint is that between the syntactic and the semantic layer there is a bijection (one-to-one and onto). He shows then that the class of number-theoretic functions which are computable relative to every notation is too narrow, containing only rather trivial functions, and that the class of number-theoretic functions which are computable relative to some notation is too broad (containing, for example, every characteristic function [page 15]). Since these constraints does not single out any standard notation, Shapiro introduces human factor: "under normal circumstances, a person engaged in computation is not merely following an algorithm. It is usually important, in particular, that the computist know the number-theoretic goal of the algorithm" [page 18].

4.4 Model-realistic solution: Model-theoretic entwining

Some Skeptics could feel disappointed by lack of a purely formal solution. What Shapiro proposes is, in the end, human-based and handwavy. If this is the case, the Skeptic might find some peace in a solution recently proposed by [9]. Dean develops a model-theoretic realism for the concept of computation. He claims that there is no point in trying to find external arguments to distinguish between various standard and non-standard models of arithmetic, nor of recursive theory. We should rather use the richness of the model-theoretic universe for studying structural properties of the concept of computability.

5 The darkest shade of the problem: Computations happen on abstract objects

There is finally another type of Skeptical worry. In his paper, [15] puts forward the idea that a full account of computability necessitated to define both syntactic and semantic computability. He then formulates a "crucial lacuna" indicating the intrinsic impossibility of defining computability on inscriptions first, and then, on its basis, computability on abstract numbers.

The lacuna states that in a realistic picture, when computations are defined on inscriptions, that there is no non-circular way of defining what computability on natural numbers is, if we want to take computability as first applying it to strings of characters. Let me remind you, that in an epistemologically plausible picture, computability on numbers as abstract objects is defined via notation. In an "epistemologically plausible picture" abstract objects are approached via language, and not via a private insight.

Rescorla's objective is to give an account of what a number-theoretic computability is. He works under three hypotheses, first, computability refers to numbers via notation (via numerals and with help of denotation function); second, Turing Machines manipulate syntactic entities; third, to specify which number-theoretic function a Turing Machine computes, one must correlate these syntactic entities with numbers. The problem is that the correlation must itself be computable, otherwise the Turing machine would compute uncomputable functions. And the circularity arises: if we propose the intuitive notion of computable relation between syntactic entities and numbers, then our analysis of computability is circular.

In consequence, Rescorla claims that computability needs to be defined as a property of abstract objects and shall be defined as such. Computability on abstract objects is defined via Church's thesis with the axioms of the theory of recursivity. This is where a Skeptic can be consoled again by the model-theoretic entwinement proposed by [9].

5.1 Moderate realism solution: any old ω-sequence will do after all

In [3] – sequel to the famous [1] – the author takes a structuralist position and claims that abstract objects playing the role of natural numbers in the structure

do not need to form a recursive progression. According to Benacerraf, there is no reason to proclaim computability of the series of abstract entities. This is so, because it is always possible to enumerate these entities with a recursive series of names. But, and here we get back to the beginning, how can we know which sequences of numerals are actually recursive...

6 Conclusions

There is no final answer that will fully satisfy our Skeptics. Each analysis of the concept of computation ends up in a vicious circle, it has a conceptual fixed point and suffers from a diagonal problem. We should keep that in mind when attempting to define computation and its twin concept of natural number.

References

1. Benacerraf, P.: What numbers could not be. Philosophical Review **74**(1), 47–73 (1965)
2. Benacerraf, P.: Mathematical truth. Journal of Philosophy **70**(19), 661–679 (1973)
3. Benacerraf, P.: Recantation, or: Any Old ω-Sequence Would Do After All. Philosophia Mathematica (4), 184–189 (1996)
4. Boolos, G., Burgess, J., Jeffrey, R.: Computability and Logic. Cambridge UP (2007)
5. Button, T., Smith, P.: The Philosophical Significance of Tennenbaum's Theorem. Philosophia Mathematica **20**(1), 114–121 (2012)
6. Carnap, R.: Foundations of Logic and Mathematics. The University of Chicago Press (1939)
7. Clarke-Doane, J.: What is the benacerraf problem? In: Pataut, F. (ed.) ruth, Objects, Infinity, Logic, Epistemology, and the Unity of Science. Springer (2016)
8. Copeland, J., Proudfoot, D.: Deviant encodings and Turing's analysis of computability. Studies in History and Philosophy of Sciences **41**, 247–252 (2010)
9. Dean, W.: Models and Computability. Philosophia Mathematica (2013)
10. Goodman, N., Quine, W.V.: Steps toward a constructive nominalism. Journal of Symbolic Logic **12**(4), 105–122 (1947)
11. Halbach, V., Horsten, L.: Computational Structuralism. Philosophia Mathematica **13**(2), 174–186 (2005)
12. van Heuveln, B.: Emergence and Consciousness. Ph.D. thesis, Binghamton University (2000)
13. Piccinini, G.: Physical Computation: A Mechanistic Account. Oxford UP (2015)
14. Quinon, P., Zdanowski, K.: Intended Model of Arithmetic. Argument from Tennenbaum's Theorem. In: Cooper, S., Loewe, B., Sorbi, A. (eds.) Computation and Logic in the Real World, CiE Proceedings (2007)
15. Rescorla, M.: Church's Thesis and the Conceptual Analysis of Computability. Notre Dame Journal of Formal Logic **48**, 253–280 (2007)
16. Shapiro, S.: Acceptable Notation. Notre Dame Journal of Formal Logic **23**(1), 14–20 (1982)
17. Tarski, A.: The concept of truth in formalized languages. In: Tarski, A. (ed.) Logic, Semantics, Metamathematics, pp. 152–278. Oxford UP (1936)
18. Turing, A.: On computable numbers, with an application to the entscheidungsproblem. Proceedings of the London Mathematical Society **42**(1), 230–265 (1936)