Recursion, Writing, Iteration. A Proposal for a Graphics Foundation of Computational Reason

Luca M. Possati

Recursion, Writing, Iteration

A Proposal for a Graphics Foundation of Computational Reason

Luca M. Possati

In this paper we present a set of philosophical analyses to defend the thesis that computational reason is founded in writing; *only what can be written is computable*. We will focus on the relations among three main concepts: recursion, writing and iteration.

The most important questions we will address are:

- What does it mean to compute something?
- What is a recursive structure?
- Can we clarify the nature of recursion by investigating writing?
- What kind of identity is presupposed by a recursive structure and by computation?

Our theoretical path will lead us to a radical revision of the philosophical notion of identity. The act of iterating is rooted in *an abstract space* – we will try to outline a *topological description* of iteration. Writing and computation are two different interpretations of this imaginary use of space. So I want to propose a minimal philosophical account[1] of computational reason.

One of the main guiding ideas of this paper is that we need new philosophical tools to understand the roots of computation and computer science. It would be useful to apply mathematical models to express and clarify philosophical problems, but this would entail major consequences for our formulation of those problems.

I will proceed as follows. The first part will be about recursion and the

1 *Philosophical* account: I will try to formulate some philosophical statements (especially in the third part). They don't ask proof or demonstration or rigorous definition. Philosophy is, to my mind, not a kind of analysis of language or a phenomenological description (*réflexion* or interpretation) of existence, but *an autonomous way of using the natural language*, a set of propositions. This makes possible the application of other tool (as mathematical models) to describe and account philosophical propositions. I will elucidate this point later.

rule-following paradox elaborated by Kripke. I will give some basic definitions of recursion and algorithms, and I will present the paradox. The second part will treat the connection between recursion and writing from the point of view of the French philosophy of technology elaborated by Bruno Bachimont. The third part will focus on iteration and will include a formal definition of iteration.

I will place my analyses at the intersection of many different theoretical perspectives and philosophical approaches: computability theory, philosophy of logic, extended mind thesis, the French philosophy of technology, philosophy of writing and Husserlian phenomenology.

1. Recursion

What can be computed in principle? In the 1930's, many mathematicians from around the world invented precise, independent definitions of what it means to be computable. Alonzo Church defined the Lambda Calculus, Stephen Kleene defined formal systems, Markov defined the Markov algorithm, and Emil Post and Alan Turing defined abstract machines to give a rigorous definition of the intuitive notion of computation. All these models are exactly equivalent: anything computable by Lambda calculus is computable by a formal system or by a Turing machine[2].

Recursive functions are another model of computation formulated by Kleene and Kurt Gödel. The *Stanford Encyclopedia of Philosophy* gives the following definition of a recursive function: "In its most general numerical form the process of recursion consists in defining the value of a function by using other values of the same function,"[3] and we can discern six basic kinds of recursion: iteration, primitive recursion, primitive recursion with parameters, course-of-value recursion and double recursion. From this definition, we can already understand the peculiarity of the recursive function: it's a function for which we have a method to find the value for all its arguments.

In their handbook, Boolos, Burges and Jeffrey write: "A function f from positive integers to positive integers is called *effectively computable* if a list of instructions can be given that in principle makes it possible to determine the value f(n) for any argument n."[4] And "the instructions must be so definite and explicit that they require no external sources of information and no ingenuity to execute."[5]

Defining recursive functions in his 1930-1931 works[6] Gödel starts with three very simple functions, called initial functions, and with two natural closure operations, composition and primitive recursion, which take some already

2 For an introduction to computability theory, see E. S. Roberts, *Thinking Recursively,* New York, John Wiley & Sons, 1986; R. Adams, *An Early History of Recursive Functions and Computability from Gödel to Turing*, Docent Press, Boston, 2011; B. J. Copeland, C. J. Posy, O. Shagrir (dir.), *Computability: Turing, Gödel, Church, and Beyond*, The Mit Press, London, 2013; M. Frixione, D. Palladino, *Funzioni, macchine, algoritmi. Introduzione alla teoria della computabilità*, Roma, Carocci, 2004.

3 P. Odifreddi and S. Barry Cooper, "Recursive Functions", *The Stanford Encyclopedia of Philosophy* (2012 Edition), Edward N. Zalta (ed.), URL = <http://plato.stanford.edu/archives/fall2012/entries/recursive-functions/>.

4 G. Boolos, J. P. Burges, R. C. Jeffrey, *Computability and Logic*, Cambridge, Cambridge University Press, 2007 (fifth edition), p. 23.

5 Ibid., p. 63.

6 See K. Gödel, "The Completeness of the Axioms of the Functional Calculus", in J. Van Heijenoort (ed.), *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*, Harvard University Press, Cambridge, 1967, p. 582-591; Id., "On Formally Undecidable Propositions of Principia Mathematica and Related Systems I", in J. Van Heijenoort (ed.), *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*, cit., p. 592-617.

defined functions and use them to define a new one. So, consider the functions f from N' to N for r (argument) = 0,1,2... We can describe the initial functions in this way:

- ζ, the zero function for the argument 0, $\zeta(\,) = 0$;
- η, the identity function for the argument 1, $\eta(n) = n$;
- σ, the successor function for the argument 1, $\sigma(n) = n + 1$.

A recursive function is a function that can be defined by these three basic functions. We call this process *inductive definition*. Recursion is a kind of mathematical induction.

In general, first we compute the value of the function for the argument zero (called the "basis" of definition); then, we suppose we know the value for a generic argument n and use it to compute the value of the *successor* of n (the inductive step). So a function is recursive iff it's a basic function (zero, successor or identity) or it can be obtained by the application of the operations of composition and primitive recursion on the basic functions. These operations "preserve" the "natural" computability of the initial functions. By applying these operations to the initial functions, it's possible to obtain an entire class of computable functions. Hence we can "define the primitive recursive functions to be the smallest class of functions that contains the initial functions and is closed under composition and primitive recursion."[7] Gödel uses the recursive functions to encode and represent mathematical language. In fact, we can convert the main mathematical operations into a recursive function.

Here are a couple of examples of arithmetic functions which are primitive recursive:

• Define the addition function, F(x,y) as follows:

$$F(0,y) = \eta(y)$$
$$F(n + 1,y) = \sigma(F(n,y))$$

• Define the multiplication function, T(x,y), as follows:

$$T(0,y) = \zeta(\,)$$
$$T(n + 1,y) = F(T(n,y), y)$$

So the primitive recursive functions are precisely these functions obtained from the initial functions by means of composition and primitive recursion. But the set of primitive recursive functions doesn't coincide with the set of recursive functions. The primitive recursive functions don't include all functions computable in principle. Gödel defined the set of recursive functions by applying new operations to the primitive recursive functions. With this definition, the set of recursive functions is exactly the same as the set of functions computable by Church's Lambda calculus, Kleene's formal system,

7 N. Immermann, "Computability and Complexity", *The Stanford Encyclopedia of Philosophy* (2016 Edition), Edward N. Zalta (ed.), URL = http://plato.stanford.edu/entries/computability/.

Markov's algorithms and Turing machines.

The *Fibonacci sequence* is a kind of recursion: a course-of-value recursion. This recursive function uses two values from previous arguments. The following provides the definition of the sequence:

$$f(0) = 0$$
$$f(1) = 1$$
$$f(n + 2) = f(n) + f(n+ 1)$$

Note the use of the two values f(n) and f(n + 1) in the definition of f(n + 2), which makes this a course-of-value recursion.

This is the process to define recursive functions. Now let's focus on the *peculiarity* of this kind of functions – the *dynamic* of the development of a recursive function.

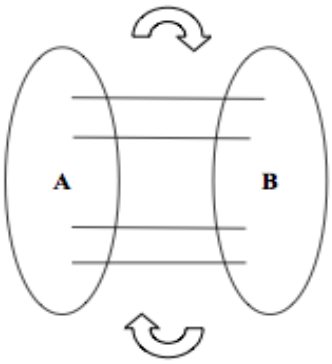
As we said, a recursive function is a function in which the values are established by using other values of the same function as arguments. These functions "are characterized by the process in virtue of which the value of a function for some argument is defined in terms of the value of that function for some other (in some appropriate sense 'smaller') arguments, as well as the values of certain other functions."[8] Therefore a recursive function "twists" in the sense that it continually changes its orientation. It's a loop, a spiral. Let's imagine a perfect map that contains a copy of itself. This is a recursive structure: there is a coincidence between the whole and the parts. It's a structure that expands by re-producing itself. We can also think about Hausdorff dimension and Mandelbrot fractals (figures that can be built by copies of themselves). This is the phenomenon of *nesting* as we can see in Escher's paintings too.[9]

First the function goes from the set A (arguments) to the set B (values), then it changes direction, and goes from the set B (arguments) to the set A (values). This change of orientation is possibile because the two sets involved in the function are always *the same set*, the set of natural numbers. There is a specific torsion of a recursive function: the sets are the same, but the direction changes. It's a "mirror function". The process is controlled by the loop, a self-reference. And what allows the loop and the inversion of direction of the function is the identity of natural numbers, so the iteration of this set.

Hence the recursive torsion from A to B or from B to A presupposes the iteration of the set (the natural numbers), that is sets A and B are copies of each other; in reality, they are always the same set. All the work to define a recursive function is nothing other than the gradual transformation of this function into other very simple functions (the initial functions) that describe this torsion, this iteration of the natural numbers that we express by means of this picture:

8 P. Odifreddi and S. Barry Cooper, "Recursive Functions", cit., p. 2

9 See D. Hofstadter, *Gödel, Escher, Bach: an Eternal Golden Braid,* New York, Basic Books, 1979, p. 127.



So *iteration*. This is the point. In classical recursive theory, iteration is defined as "the simplest type of recursion" that "occurs when a given function is iterated."[10] And "*it's possible to prove that any primitive recursion can be reduced to an iteration, in the presence of a coding and decoding mechanism.*"[11] So we can claim that iteration is *the core of recursion*.

The role of iteration as the basis of an alternative foundation of mathematics opposed to set theory is underlined by Church[12]. Church's idea is to represent the natural number *n* as the binary operator $\overline{n}$, that, when applied to the arguments f and x, produces the *nth* iteration; every natural number can be written as follows: $f^{(n)}(x)$. The number is an *iterator*. We can find the same idea in Wittgenstein and Peano's axioms.[13]

But, if iteration is the core of recursion, what is iteration? Is there a logic of iteration? A number is a *pure* iteration. But, can a *pure* iteration exist? How can we think of something as a *pure* iteration? The concept of a *pure* iteration involves the multiplication of identicals: this a contradiction. These questions are the crucial point of our inquiry, but I will return later.

Now I want to focus on another important point. We have to notice that classical recursion theory entails a strong philosophical paradox called Kripkenstein paradox[14], as it has been formulated in Kripke's *Wittgenstein on Rules and Private Language*. Kripke's Wittgenstein (the Kripkean interpretation of Wittgenstein) has become a philosopher in his own right, and for many people it is not an issue whether the historical Wittgenstein's original ideas about private language are faithfully captured in this version. But, for our purposes, we take the paradox to aim to the connection between the different steps of inductive definition. Kripkenstein prevents us from admitting the classical definitions of recursion and algorithms. I want to use the sceptical power of the paradox.

Let's consider the Kripkenstein paradox. A preliminary remark: the debate over Kripke's argument is too vast to survey here. Limitations of time

10 P. Odifreddi and S. Barry Cooper, "Recursive Functions", cit., p. 6.

11 Ibid., p. 9. See P. Odifreddi, *Classical Recursion Theory*, volume I, Amsterdam, North Holland, 1999 (second edition).

12 See A. Church, "A set of postulates for the foundation of logic" (first paper), *Annals of Mathematics*, 1932 (33), p. 346-366; Id., "A set of postulates for the foundation of logic" (second paper), *Annals of Mathematics*, 1933 (34), p. 839-864.

13 See L. Wittgenstein, "Logisch-philosophische Abhandlung", *Annales der Naturphilosophie*, 1921, 14, p. 185-262; G. Peano, "Sul concetto di numero", *Rivista di matematica*, 1891, p. 87-102, and 256-267. See also M. Mathieu, *Wittgenstein. Finitism and the Foundation of Mathematics*, Oxford, Clarendon Press, 1998.

14 See S. Kripke, *Wittgenstein on Rules and Private Language*, Harvard, Harvard University Press, 1982.

dictate that a choice be made, and we are interested only in the philosophical assessment of the paradox considered on its own terms and independently of exegetical questions. So I don't want to establish the correctness of Kripke's interpretation or to analyse all the literature about it. I want just to focus on the sceptical core of the paradox and its implications for computation.

In 201 paragraph of *The Philosophical Investigations* Wittgenstein writes: "This was our paradox: no course of action could be determined by a rule, because every course of action can be made out to accord with the rule […]." According to Kripke, "the impossibility of private language emerges as a corollary of [Wittgenstein's] sceptical solution of his own paradox."[15] Kripke takes the paradox to pose a genuine and profound sceptical problem about meaning. Every new application of a rule is a leap in the dark because the rule doesn't contain the criterion of its application and interpretation. There is no fact that can determine what the rule means and the conditions of validity of its use. It's the same idea at the core of Quine's argument about translation: there is not an objective criterion to determine the reference of our words[16].

The example Kripke chooses to illustrate the problem is that of addition. What is it to grasp the rule of addition? The fact that 68 + 57 yields 125 – the consequence of the operation denoted by the sign "+" – doesn't depend on a rule or a series of rules. Let's consider the set of numbers that we have never added before. I assume that 68 and 57 belong to this set: I have never used the operation "plus" for them. Then, when I add 68 and 57, I obtain 125. Nevertheless, a sceptical thinker could ask me: "When you added two numbers in the past – two numbers less than 68 and 57 – by "+" you always meant not the operation you call addition, but another one called *quaddition* or *quus*. Quaddition is an operation completely identical to addition except that it yields 5 instead of 125 in the case of 68 + 57 (because, for example, only for this case it could presuppose a subtraction: 125 – 120, and I cannot recognise it). Adding 68 + 57 for the first time, you believe you should perform the same operation as in the past, but it's an illusion".

The point is that there is no fact in my previous experience that can contradict the sceptical thinker. I cannot prove that when I used the sign "+" in the past, I was not performing a quaddition but an addition. I cannot determine whether 68 + 57 yields 125 (using addition) or 5 (using quaddition). *Is "+" addition or quaddition?* Nothing in my previous experience helps me to answer this question because all my previous uses of "+" were equally compatible with addition and quaddition.

So what is the meaning of "+"? Kripke says: "The sceptic argues that when I answered '125' to the problem '68 + 57', my answer was an unjustified leap in the dark; my past mental history is equally compatible with the hypothesis that I meant quus, and therefore should have said '5'."[17]

Let's express the paradox in this way. I have a mathematical problem X, a set of rules and strategies C to solve this problem, and a set of conditions B that the rules and the strategies have to observe to be correct. Kripke says that we

15 Ibid., p. 13.
16 See W. V. Quine, *Word and Object*, Mit Press, Cambridge, 1960.
17 S. Kripke, *Wittegenstein on Rules and Private Language*, cit., . p. 15.

have not one set of rules for X, but many. There are many sets of rules and strategies that observe the B conditions and could be used to solve X, but are incompatible – they contradict each other. We choose only one of these sets for practical reasons, but our choice is arbitrary. The point is that there is no fact either in X or in B that justifies the use of one of these sets instead of another. There is not a unique meaning of X and a unique solution. As Quine proved, meaning is always a relative notion.

Let me clarify the implications of the sceptical argument. It doesn't concern mathematical technique, but it has disastrous consequences for computation.

In particular, it disrupts the consistency between my present and past uses of "+". I cannot prove that 68 + 57 yields only 125 and not 5, and that I have to compute only in this way. If an algorithm is a set of operations, the Kripkenstein paradox questions the unity of this set and its temporal development. Computation can always explode at every step. Kripke says: "Perhaps when I used the term 'plus' in the past, I always meant quus: by hypothesis I never gave myself any explicit directions that were incompatible with such a supposition. […] no fact about my past history – nothing that was ever in my mind, or in my external behavior – establishes that I meant plus rather than quus"[18]. The paradox invalidates mathematical induction[19]. Why is the numerical system we use continuous?

How can we reply to this paradox? There are many attempts to elucidate the deep structure of the paradox and to solve it. For a survey, you can see Boghossian's paper[20] – a very meaningful synthesis.

I don't want to propose another analysis, account, or solution of the paradox. I want to learn from the paradox. I want just to examine its consequences for computability. *How does the crisis of meaning undermine computation?*

According to Kripkenstein, the coherence of recursion and induction is not based on the rules we use and their meanings. Nevertheless we can compute. In a Turing machine every formal rule corresponds to a different movement of the machine (the head on the tape). There is no meaning, disposition, intention, or interpretation that guides the machine. A rule is a movement on the tape and a state of the machine. So what allows us to talk about a coherent process? It is not the hidden meaning or the particular symbolism that we use. What supports computation?

I propose to look at writing itself, the graphics structure of a formula. Writing is a mediation between (1) a mathematical problem, (2) our operations to solve it, and (3) any other facts involved (rules, conditions, behaviour, etc.).

Focus on this point. According to the French philosophy of technology, writing is not a neutral tool, but *a material a priori* that influences and shapes our rationality. Writing supports and guides the development of computation and our related dispositions. The Kripkenstein paradox exists only when we think of intentionality as something isolated and dictinct from the technology it uses. But

18 Ibid. p. 13.
19 Ibid. p. 18-19.
20 P. A. Boghossian, "The Rule-Following Considerations", *Mind*, 1989 (98), p. 507-549.

intentionality without technology doesn't exist. Our intentionality – our "will" to meaning – is always driven and guided, not determined, by technology. And writing is a tool, a technique, a technology. Technical facts drive, guide and orient dispositions. The coherence of computation is preserved by writing. After Kripkenstein and the fall of meaning, comes Derrida and what Derrida calls the *graphosphère*, "la forme papier du savoir"[21]. This also opens new perspectives on language and rationality.

2. Writing

In the second part of this paper, we present the general thesis that writing is the *material* a priori of computation. This is not a new kind of formalism. The fall of meaning due to the Kripkenstein paradox forces us to consider seriously the connection between computability and writing.

By "writing" I mean a very general concept. As anthropologist Jack Goody shows, writing has had a profound influence on the development of our cognitive abilities. In his book *The Domination of the Savage Mind*[22], Goody proves that societies that write think differently from those that cannot. Writing entails specific cognitive operations: a kind of reason that he calls *graphical reason*. Goody claims that writing produces three main conceptual structures: lists, tables and formulas.

However, writing can also have a wider philosophical meaning. The most important thinker who recognised the philosophical dimension of writing is Jacques Derrida. From a phenomenological perspective, Derrida affirms that experience understood as experience of the present is never a simple experience of something present over and against me, right before my eyes as in an intuition. There is neither a pure presence nor a pure absence, neither a pure identity nor a pure difference, neither pure culture nor pure nature. Experience is always a blend of presence and absence, identity and difference, culture and nature. In *De la grammatologie*[23], Derrida calls this structural contamination *différance* – a structural break that characterizes all experience. Writing as supplement, deferment, replacement expresses this *différance* and is the origin of language. Writing is the *trace* that founds all experience. Writing is fundamentally iteration or – as Derrida says – *iterability*. If the present is always complicated by non-presence, iterability contains what has passed away and is no longer present, and what is about to come and is not yet present. Something could be an experience iff it is repeatable. There must be a minimal iterability found in every experience.

Bruno Bachimont develops the positions of Goody and Derrida. The main thesis of Bachimont's philosophy of technology is that technology shapes our rationality. We cannot treat knowledge and rationality without considering the technical factor inside both. Mind is influenced by machines built by the mind itself. And the most important machine is writing. Writing is the essence of

21 J. Derrida, *Papier machine*, Galilée, Paris, 2001, p. 248.

22 J. Goody, *The Domestication of the Savage Mind*, Cambridge, Cambridge University Press, 1977. For the anthropology of writing see also: A. Le Roi-Gourhan, *Le geste et la parole*, I-II, Paris, Albin Michel, 1964; Id., *Milieu et technique*, Paris, Albin Michel, 1973.

23 J. Derrida, *De la grammatologie*, De Minuit, Paris, 1967.

every technology.

Bachimont writes: "Se dégage l'idée que nos outils intellectuels, selon leurs nature et leurs propriétés, nous aident à penser différemment, comme les outils mécaniques permettent de réaliser des objets matériels différents. Et de la même manière qu'il existe une histoire des techniques et des objets qu'elles permettent de réaliser, il existe une histoire de nos outils intellectuels et des modes de pensée qui y sont attachés."[24]

Bachimont thinks of computational reason and computer science as evolutions of graphical reason. Starting from Goody's analyses, he distinguishes three main structures of computational reason: programs (software), nets (relations among programs) and layers (hardware)[25]. So writing is the *a priori* that computation needs to build itself. And this a priori is *material* because it entails also a modification of matter. Writing is a mark on a material support. This act makes computational processes possible. It is a *technical a priori*.

We do not have time to treat the details of this theory here. I will just note that, according to Bachimont, there is a direct correspondence between graphical reason and computational reason that we can express as follows:

WRITING	Cognitive operations	**COMPUTATION**	Cognitive operations
Lists	Classify	**Programs (Software)**	Systematic set of procedures
Tables	System	**Nets**	Communication in the system
Formulas	Use of a pure form to reason	**Layers (Hardware)**	Codes to convert in machine

How can writing produce its conceptual structures? In a Kantian way, Bachimont thinks that writing is a space-time synthesis – he says: *une synthèse synoptique de l'écriture*. Writing is a synthesis between permanence in time and space. It synthesizes, on the one hand, memory (the need for unity and order in the temporal flow) and, on the other, a kind of organization based on simultaneity. In writing we have a *spatialisation of time*. The temporal development of discourse (linguistic acts in general) is translated into a spatial organization, into a simultaneous order: in lists, tables and formulas, the items are defined by their position in an abstract space. Writing gives us a synthetical way to grasp together dispersed items in the temporal flow. Lists, tables and formulas are the main graphics schemes by which writing can realize this space-time synthesis. The same is the case for computation. These graphical schemes mark and modify matter. From the point of view of technology, this is their *telos*.

This is the position of French philosophy of technology. Now I want to

24 B. Bachimont, "Signes formels et computation numérique: entre intuition et formalisme" (à paraître).

25 See B. Bachimont, *Le sens de la technique: le numérique et le calcul*, Paris, Encre Marine, 2010; Id. "L'intelligence artificielle comme écriture dynamique: de la raison graphique à la raison computationnelle" in J. Petitot, P. Fabbri (dir.), *Au nom du sens*, Paris, Grasset, 2000, p. 290-319. See also B. Stiegler, *La technique et le temps*, I-II, Paris, Galilée, 1994.

try to improve on Bachimont's views. My position is that before lists, tables and formulas, there is another graphical writing scheme: the character. This is the simplest scheme. In order to write, we always need a set of characters, basic images that combine with each other iteratively. So character express the minimal iterability found in every experience.

Consider the following four points:

- I will call a "type machine" the set of characters and their ability to be repeated and combined in different ways. A type machine is an abstract machine: a set of items that can be iterated and moved iteratively.

- The operation that corresponds to a character is iteration. Characters work by iteration. It is an indisputable fact that a character is an item that I can use only by iterating it.

- A character is not a symbol: I mean that a character is always linked to the other graphical schemes of writing (lists, tables, formulas) and it is iterated in terms of a typographical structure. A symbol presupposes a character; a symbol is always a character. But a character is not necessarily a symbol.

- Iteration is not a space-time synthesis, but the intuition and the construction of an abstract space. For this reason, I suggest the possibility of a topological model to describe iteration – a proposal for a philosophical interpretation of *topology*[26].

The Turing machine is essentially a type machine. In his landmark 1936 paper, *On Computable Numbers*, Turing writes: "We may compare a man in the process of computing a real number to a machine which is only capable of a finite number of conditions *q1, q2.....qr*, which will be called '*m*-configurations'. The machine is supplied with a 'tape' (the *analogue* of paper) running through it, and divided into sections (called 'squares') each capable of bearing a 'symbol'. At any moment there is just one square, say the *r-th*, bearing the symbol Σ (r) which is 'in the machine'. We may call this square the 'scanned square'. The symbol on the scanned square may be called the 'scanned symbol'. The 'scanned symbol' is the only one of which the machine is, so to speak, 'directly aware'. However, by altering its m-configuration, the machine can effectively remember some of the symbols which it has 'seen' (scanned) previously."[27]

Turing has a typographical conception of his machine: symbols don't refer to a meaning, but to a position on the tape. We have to focus on the difference between character and symbol: a character is always linked to the other graphical schemes of writing (lists, tables, formulas), while a symbol is independent of those.

Turing writes: "If an a-machine *prints* two kinds of symbols, of which the

26 For an introduction to topology. see B. Mendelson, *Introduction to Topology*, London, Dover, 2012 (third edition); S. E. Goodman, *Beginning Topology*, London, Orient Black Swan, 2012; M. Manetti, *Topologia*, London-Milano, 2014.

27 A. M. Turing, "On Computable Numbers, with an Application to the *Entscheidungsproblem*", *Proceedings of the London Mathematical Society*, 42, 1936-37, p. 231.

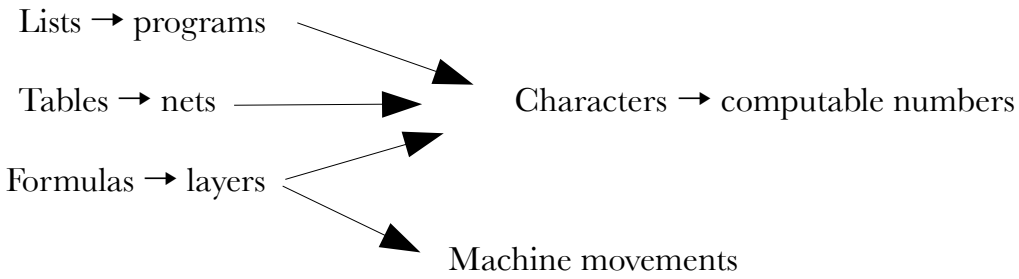
first kind (called figures) consists entirely of 0 and 1 (the others being called symbols of the second kind), then the machine will be called a computing machine"[28]. In a Turing machine each symbol and each sequence of symbols is constructed by the tape, that is the typographical structure of the machine itself. Turing uses the term "symbol" to refer to what I have called character. There is only the iteration of the symbols (1,0) on the tape, according to the instructions. I'm not deleting the semantic and interpretative dimension. I'm just saying that the most basic condition of computation is an iterative structure.

So, following Bachimont's perspective, lists make programs possible, tables make nets possible, formulas make layers possible. Then, we can say that characters make computable numbers possible. Every computable number can be seen as an iterator. Computable numbers derive from the natural iterability of writing. In other terms, a number is computable because it can be written.

Hence we can also say that a computable number is a character that is iterated following a program (a set of procedures, an algorithm) in a net (a systematic set of other characters and procedures) and according to some codes that convert this net into a series of movements of a physical machine (a material support, hardware). A number that cannot be computed is an iteration that cannot be linked to a program, to a net or converted by a code into a series of physical movements.

I'm not saying that numbers are abstractions from characters – as if the zero would be an abstraction from the letter "o"! – but that writing is the most basic condition of the ability of numbers to be computed. Numbers are iterations: they have an iterative nature. This iterative nature can be computed iff it can be convert in writing and "checked". But writing cannot exhaust this iterative nature.

This is a kind of transcendental deduction – to use a Kantian term – of computation, *not of numbers in general*.



Let's summarize our results:

1. The Kripkenstein paradox forces us to focus on the typographical structure of mathematical computation;

2. This typographical structure is not a formal mechanism, but includes a series of graphical schemes (space-time syntheses);

3. Computable numbers derive from the natural iterability of writing.

28 Ibid.

The program of a transcendental deduction of computation entails:

1. A precise definition of graphical structures of writing and computation;

2. A precise definition of the correspondence between these structures;

3. A precise definition of the idea that graphical schemes are space-time syntheses as a work of social imagination.

3. Iteration

What is the sense of our transcendental deduction? Recursion and writing point back to iteration. But – following a suggestion of Derrida – iterability cannot be understood in terms of classical logic. So in this section, two questions will concern us:

- What exactly is iteration?
- Can we elaborate a logical definition of iteration?

I will organise my analysis in three steps.

1. First step: *identity of number as object*. Consider an iterated function

$$f(x) = x$$

that is a function from some set to itself, a certain number of times. This kind of function – as we said – is a very powerful tool in mathematics and computer science, with many applications.

However, if we admit the Kripkenstein paradox, how can we admit also the possibility of an iterated function? How can mathematicians say that x and x (two symbols in the formula) designate the same set? How can mathematicians be sure that it is always the same set and that, for instance, the first set (first designation) is not larger than the second one (second designation)? How can mathematicians be sure that the set has not changed and grown?

If we admit the Kripkenstein paradox and we want to maintain identity, we have to admit also the possibility of *iteration*: that a number is always the same from the state 1 to the state 2, and for all possible states *n*. A number cannot change through time. A number has to be wholly present in every possible state. If we admit the Kripkenstein paradox, we have to affirm that a natural number cannot have different temporal parts, but only spatial parts in a simultaneous order – the simultaneous order of the formula. This is a metaphysical thesis: natural numbers endure, not perdure[29]. The Kripkenstein

29 For the difference between *perdurance* and *endurance*, see H. Noonan and B. Curtis, "Identity", *The Stanford Encyclopedia of Philosophy* (2014 Edition), Edward N. Zalta (ed.),

paradox teaches us another crucial point.

As a result, we can say:

1) a number is not an ordinary object, but an object *sui generis*;

2) this object is defined only by its structural properties – a number has only a positional identity;

3) this structure is an iterative structure – the status of a number entails the possibility of multiplication or reproduction of identicals, that is a pure series of *identical* copies.

2. Second step: *identity of number as structure*. The sequence of natural numbers can be described in the following way: we start from zero and then we iterate the successor operation (we add a unity). We can depict it as follows (→ designates iteration):

$$0 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow n$$

Every number is different from every another one. But the successor operation is always the same and constructs the sequence. My question is: what is the logic of →?

Consider the nature of numbers and the relation between numbers and sets. According to Boolos's iterative and cumulative conception of set theory, iteration leads always to the next level of the hierarchy of sets, and every level represent a set and so a number[30].

Nevertheless, as Benacerraf demonstrates[31], this way of thinking is wrong. The main mistake of set theory is to consider numbers and sets to be objects with many properties, and this conviction leads to different and contradictory representations of numbers. In fact, these representations observe different or contrary axioms. But, as Benacerraf says also, a number has only structural properties, nothing else. The nature of number itself is trivial, indifferent, empty. 1 is only a place in a structure. Mathematics studies the structures in which numbers designate positions or places, and not the nature of numbers. A number is defined by structure, and the most basic mathematical structure is iteration. A number has only structural or positional properties.

Consider the Mandelbrot fractal. We can consider the fractal from different levels of proximity – from different points of view. In other terms, we can go down in its deeper levels proceeding by degrees, to infinity. Consider the following phenomenon: a part of the fractal that appears different from another one at level a, can coincide with it at level b, and vice versa. The whole coincides with the parts. The same is true for numbers. Numbers are only different places or points of the same general iterative structure. In this structure, every point can coincide with another one, to infinity. The different structures by which we consider numbers are the degrees of fractal. But a number can also be seen in itself as an iterative structure with some places or

URL = http://plato.stanford.edu/archives/sum2014/entries/identity/.

30 See G. Boolos, "The Iterative Conception of Set", *The Journal of Philosophy*, 1971 (68), p. 215-231.

31 See P. Benacerraf, "What Numbers Could not Be" (1965), in P. Benacerraf, H. Putnam (eds.), *Philosophy of Mathematics: Selected Readings*, Cambridge University Press, Cambridge, 1983, p. 272-294. See also S. Yablo, "Go Figure: A Path Through Fictionalism", *Midwest Studies in Philosophy*, 25, 2001, p. 72-102; S. Yablo, "Abstract Object: A Case Study", *Noûs*, 3, 2002, p. 220-240.

parts.

So, back to the question: what is the logic of →? I add a unit. Nothing else. But what is this unit? A unit is a place in a structure. A unit has only a positional or structural identity. So when I add a unit, I'm linking a place to another one in a structure. I'm building a topology. The units are empty. They are different *only* in their positions. Their distinctness and identity depend *only* on space.

3. Third step. By referring to numbers and their iterative nature, we have obtained a minimal condition of iteration. An iterative structure is a structure in which distinctness and identity of items that compose it depend *only on space*. It is a minimal and negative kind of identity. An iterative structure has only one fundamental restrictive condition: every unit has only structural properties. A temporal order necessarily adds another properties, as before and after, to structural properties. When I iterate an object (physical or not), I think of this object as an iterative structure, and so I abstract from its concrete nature and properties. An iterative structure is another kind of reflexivity, different from classical models of identity (indiscernability of identicals and identity of indiscernables).

French philosopher Cornelius Castoriadis can help us to elucidate the link between iteration and space. Castoriadis thinks of space – he says "pure space" – as the condition of iteration. There is a "pure topology" at the roots of the act of iterating. In order to iterate this picture

I need to put an *interval*, a space, like this

I can continue in this way:

There is an extension of space: a simultaneous order is created. Always the same act, always the same result. It is not possible to distinguish the points except by their spatial-positional difference. Time cannot admit the iteration of identicals. To iterate, we need an extension of space. Here space assume a logical function.

Think about photography: we can reiterate exactly the same photo to infinity. We can have many different copies of the same photo. Those photos are not tokens of a type, because there is no difference among them. We cannot recognise a type: immediately after the production of the first copy, the difference between this first copy and the original disappears. What is the relation among the copies? They are different from each other, but always the same photo. What is the logic that sets up the production and reproduction? I have many different copies of the same photo (of the same instant) over and

against me, right before my eyes, in a simultaneous order.[32] I can make it because I have "written" this instant, fixed it on a typographical structure, and arranged in a space. But what kind of space?

In *The Imaginary Institution of Society*, Castoriadis writes:

"A 'pure' space is, from a reflexive and analytical point of view, a necessity for thought and for its most elementary operations. To think, we must be able to grasp the same as different and vice versa; we must be able, for example, to *iterate* or *repeat*, to retain as plural and as different the absolutely selfidentical that 'repeats' itself. This is the possibility that 'pure' space provides, the possibility for the 'same' point to be different if it is placed somewhere else; and in this sense, reflexively, space exists prior to the figure as its a priori condition. There is nothing like this in the case of time, which would be nothing if it were the mere possibility of iteration of the identical. An 'empty' space is a logical and physical *problem*; an 'empty' time is an absurdity – or else, it is simply a certain name given for no particular reason to a spatial dimension. What would time be if only the same existed? If I 'extend' a square or a circle in a plane into an infinite parallelepiped or cylinder, and, then, reiterate them interminably in an additional dimension, I obtain, precisely, no more than an additional spatial dimension; what I am doing is still geometry. In the same way, if I were to stretch out the 'sphere of the world' into a fourth dimension, I would still be working on the geometry of a hypersphere in R4. Even physics cannot be content with this."[33]

A "pure space" is a problem for physics. Nevertheless it is the condition of thinking. Castoriadis writes:

"'Pure' space is the possibility of difference in so far as it is the condition of the repetition of the same as different, that is to say, as iteration – atemporal repetition, in the forever, *aei*, of spatiality or of coexistence, or of composition. In its most basic form, it is what allows the possibility of granting (or of 'seeing') that points x and y are at once the same (in that nothing intrinsic to them distinguishes them) and different (in and through their spacing). As such, *it is 'logically' (and not 'psychologically') presupposed by logic and mathematics*, because it is already presupposed by the most elementary *legein*."[34]

Can we convert these statements in a topological theory ("topological" in a mathematical sense)? In Castoriadis's model of iteration, space assume a logical function. However, Castoriadis doesn't elaborate a logical account of this function. He limits himself to an exegesis of Plato's *Timaeus*. Can we elaborate this logic? Can a paraconsistent logical approach[35] help us to define iteration?

Let's follow Castoriadis again:

"If the *sign* is to exist, the different must be identical and the identical must be different or capable of differentiating itself, multiplying itself, making itself plural without ceasing to be the same. a and a are the same independently of

32 See R. Barthes, *La chambre claire. Note sur la photographie*, Paris, Gallimard, 1980.

33 C. Castoriadis, *The Imaginary Institution of Society*, tr. by K. Blamey, Cambridge, Mit Press, 1997, p. 119 (original work: *L'institution imaginaire de la société*, Paris, Seuil, 1975).

34 Ibid., p. 120.

35 See G. Priest, R. Routley, *The Philosophical Significance and Inevitability of Paraconsistency*, in G. Priest, R. Routley (dir.), *Paraconsistent Logic: Essays on the Inconsistent*, München, Philosophia Verlag, 1989, p. 483-539; G. Priest, *In Contradiction. A Study of the Transconsistent*, Oxford, Clarendon Press, 2006 (second edition); G. Priest, *An Introduction to Non-Classical Logic*, Cambridge, Cambridge University Press, 2008; G. Priest, *Doubt Truth to Be a Liar*, Oxford, Clarendon Press, 2006; G. Priest, *One. Being an Investigation into the Unity of Reality and its Parts, including the Singular Object which is Nothingness*, Oxford, Oxford University Press, 2014; G. Priest, *Logic of Paradox*, in "Journal of Philosophical Logic", 8, 1979, 219-241; G. Priest, *Logic of Paradox Revisited*, in "Journal of Philosophical Logic", 13, 1984, p. 153-179.

where they may appear on the page. And a is not a sign – letter or phoneme – if it cannot be made plural, be iterated, and if it cannot become 'different' (take on different 'values') while remaining the same and simply being in another 'position': the two '1's of the figure '11' acquire their difference in sameness due to their place."[36]

There is a work of imagination at the roots of iteration and identity. This is the imaginary *dawn* of rationality. Rationality is a moment of this work of imagination that is social. So we should find the primitive imaginary formations that make possible different kinds of recursion (iteration, primitive recursion, primitive recursion with parameters, course-of-value recursion and double recursion).

4. Conclusion

We conclude our path with some open questions:

- Can we elaborate a rigorous definition of an abstract space that allows iteration? What does the term "abstract" mean here? We have to precise our conception and our method of abstraction. This entails a consideration of contemporary new-fregean theory of abstraction[37].
- What kind of social imagination produce this abstract space?
- Can we use the concept of a type machine also to study the propositions of the natural language?
- How does the type machine (for example, our alphabet) influence the construction of predication? Is predication a kind of computation? If a predication can be thought as a function, as Frege shows, can we detect the initial functions that recursively define every meaningful predication?

36 C. Castoriadis, *The Imaginary Institution of Society*, cit., p. 120.

37 See B. Hale, *Abstract Object*, Oxford, Blackwell, 1987; B. Hale, C. Wright, *The Reason's Proper Study: Essay Towards a New-Fregean Philosophy of Mathematics*, Oxford, Clarendon Press, 2001.