

# Sussex Research

## Incomputable aesthetics: open axioms of contingency

Beatrice Fazi

### Publication date

15-01-2016

### Licence

This work is made available under the [CC BY-NC-ND 4.0](#) licence and should only be used in accordance with that licence. For more information on the specific terms, consult the repository record for this item.

### Document Version

Published version

### Citation for this work (American Psychological Association 7th edition)

Fazi, B. (2016). *Incomputable aesthetics: open axioms of contingency* (Version 1). University of Sussex. <https://hdl.handle.net/10779/uos.23428361.v1>

### Published in

Computational Culture: a journal of software studies

### Copyright and reuse:

This work was downloaded from Sussex Research Open (SRO). This document is made available in line with publisher policy and may differ from the published version. Please cite the published version where possible. Copyright and all moral rights to the version of the paper presented here belong to the individual author(s) and/or other copyright owners unless otherwise stated. For more information on this work, SRO or to report an issue, you can contact the repository administrators at [sro@sussex.ac.uk](mailto:sro@sussex.ac.uk). Discover more of the University's research at <https://sussex.figshare.com/>

**Computational Culture**

a journal of software studies

# Incomputable Aesthetics: Open Axioms of Contingency

**ARTICLE INFORMATION**

- **Author(s):** M. Beatrice Fazi
- **Affiliation(s):** Sussex Humanities Lab, University of Sussex
- **Publication Date:** 15th January 2016
- **Issue:** 5
- **Citation:** M. Beatrice Fazi. "Incomputable Aesthetics: Open Axioms of Contingency." *Computational Culture* 5 (15th January 2016). <http://computationalculture.net/incomputable-aesthetics-open-axioms-of-contingency/>.

**ABSTRACT**

In 1931, Kurt Gödel determined the incompleteness of formal axiomatic systems by demonstrating that there are propositions that cannot be proved or disproved within the system in question. In 1936, Alan Turing showed that some functions cannot be computed, and thereby described the limits of computing machines before any such machine was built. In this essay I will turn to these logical discoveries in order to argue that incompleteness and incomputability can be employed as conceptual tools to re-engage with the axiomatic character of computation. This re-engagement with formal axiomatic structures, I will claim, has important consequences for the aesthetic investigation of computation. Computational aesthetics is understood here as an enquiry into the relation between abstraction and experience in computation. In this respect, the concepts of the incomplete and the uncomputable will be mobilised philosophically, beyond the technical scope of Gödel's and Turing's work, to argue for the autonomy and reality of computational abstraction. Gödel's and Turing's discoveries preclude the possibility that axiomatic formulation could be the method through which the metacomputation of the intelligible and the sensible is accomplished. These discoveries in fact prove that the computational axiomatic system is not transcendently closed to contingency, as metacomputational approaches geared towards the formulation of reality would have it, but is instead immanently open to its own eventuality. I will thus argue that there is a contingent ontology of computation that is to be found within computation's formalisms. This contingent ontology is not predicated upon the empirical or the phenomenal, but is inherently formal and computational. For the philosophical investigation of the aesthetics of computation, such an ontology of the contingent computational structure opens up the possibility of thinking computational forms beyond the limits of formulae, together with that of engaging with formalism beyond the idealisation of beautiful and truthful determinisms.

## Prologue: Gödel and Turing

In 1936, *Proceedings of the London Mathematical Society* published a paper authored by a young Cambridge Fellow, Alan Turing. This essay, 'On Computable Numbers, with an Application to the Entscheidungsproblem',<sup>1</sup> is considered today to be momentous for having stripped what was then the purely mental activity of computation down to its basics, and for having thus proved what hypothetical computing machines could or could not achieve before actual computers were built.

Turing originally wrote the paper in an attempt to engage with some of the programmatic questions posed by the most influential mathematician of his time, David Hilbert. In the early 1920s, Hilbert proposed a research programme that was famously summarised in three key points: consistency, completeness and decidability. Hilbert believed that the increasing abstractness of mathematical reasoning called for the systematisation of all mathematics into the axiomatic form and for a general 'metamathematical' proof able to secure the provability of such a formalisation. However, in 1931 the logician Kurt Gödel showed that Hilbert's programme could not be carried out, because consistency and completeness cannot both be accomplished within the same formal system.<sup>2</sup> What Gödel's 'incompleteness theorems' famously demonstrated is that statements exist which, when viewed from the outside, are certainly true, but the provability of which cannot be deduced inside the formal system in question. Gödel's proofs concerned the problematic reality of *undecidable propositions*, whose existence can be expressed but whose decidability cannot be demonstrated through the finite axiomatic means of the formal system itself. Gödel's response to Hilbert incontrovertibly ruled out the ability to derive all true formulae from axioms. Consequently, it precluded the possibility that any final systematisation can be accomplished, or that any assurance can be given regarding the absence of internal contradictions in the foundations of mathematics.

Studying and writing in the aftermath of the incompleteness bombshell, Alan Turing approached Hilbert's foundational demands by coping directly with the need for a conclusive definition of 'method'. His paper addressed the *Entscheidungsproblem* (German for 'decision problem'), namely Hilbert's challenge concerning the discovery of a general symbolic process capable of delivering either 'true' or 'false' as output, and of doing so in conformity with the actual truth or falsehood of the formula itself. Turing proved that such an algorithm does not exist by devising a thought experiment (today known as the

'Turing machine') that was so compelling that it also indirectly proved that the reduction of all deductive reasoning to mechanical calculation cannot be accomplished. Whereas Gödel addressed the theoretical consistency and completeness of formal rules for the description of logic and mathematics, Turing investigated the possibility of finding a mechanical, finite-step and formally expressed process that would set the standard for deductive procedures. Arguably, his genius consisted in casting the problem in terms of computability. "The real question at issue", Turing wrote, "is 'What are the possible processes which can be carried out in computing a number?'"<sup>3</sup>

In attempting to answer this question, Turing's 1936 paper provided a formal method of computation, laid out through his hypothetical computing machine. This ideal device is composed of an infinite tape. The tape is divided into cells, each of which may contain only one symbol from a finite alphabet. Such symbols are written and read by a head that moves left or right on the tape, one cell at a time. The Turing machine is extremely simple: its activity consists in writing or erasing a determinate symbol, moving the head one square to the left or right and changing to a new state. Yet, the Turing machine is also profoundly complex, and is so precisely by virtue of its generality. Each set of rules controlling the movement of the head could be considered a different Turing machine in its own. However, one machine (which is therefore said to be *universal*) can be envisaged that could perform, given the proper inputs and the proper configuration, any task that any individual Turing machine could carry out. In other words: anything computable through a definite set of instructions, which can be composed as a series of symbols, can also be computed by the Universal Turing Machine (hereafter, UTM). The problems that a UTM can resolve, therefore, are all those that are executable by an algorithmic process or by an 'effective method' of computation, within any reasonable definition of these terms.

Returning now to Hilbert's *Entscheidungsproblem*: Turing proved that there is no algorithm (i.e. no effective method) that can decide, in advance of its calculation, whether a statement is true or false. He did so by demonstrating that there are some tasks that cannot be accomplished by his universal computing machines. These problems are said to be *incomputable functions* with no solution. Turing thus showed that there are limits to what can be computed, because there are functions that can be said to be algorithmically irresolvable. This, in consequence, invalidated the Decision Problem and its solubility as a whole. It also provided a mathematical model for an all-purpose computing machine, as well as formalising a very concrete notion of definite

procedure and putting forth the definition for a whole new category of numbers (the computable numbers).

## **Introduction: A Closed System?**

As the historical prologue above has indicated, Gödel and Turing offered solid explorations of the decidability of formal axiomatic systems. Their discoveries are paradoxical, but extremely powerful. Gödel showed the incompleteness of the axiomatic method by constructing a deductive system that demonstrates its own improvable. Equally, Turing showed that the formal notion of computation rests upon the conclusion that there are numbers and functions that cannot be computed. The impact of Gödel and Turing's breakthroughs had the force of a collision: it took mathematical logic by storm, whilst ushering in new philosophical, scientific and technological challenges.

In this essay, I will claim that Gödel and Turing's discoveries also carry unavoidable consequences for the ontological investigation of computational aesthetics. This is because it is possible to mobilise the incomplete and the incomputable beyond the scopes of Gödel's and Turing's original papers, in order to uncover something important about what aesthetics in computation might account for. This essay will propose an explicitly theoretical argument. The term 'aesthetics' will not be employed here in terms of a theory of art, but rather in a philosophical sense: as a theory concerning the construction of experience, or indeed as an investigation into the possibility thereof. The concept of experience, in turn, will also be addressed in a manner that extends well beyond the psychological territories of conscious human activity, and will instead be considered from an ontological perspective, in terms of self-determination vis-à-vis indeterminacy.

Generally and traditionally speaking, aesthetics concerns the sensible, and what we encounter qualitatively. From an onto-aesthetic perspective, which draws from the etymological root of 'aisthesis', actuality determines itself vis-à-vis the indeterminacy of life and sensation. My proposition does not dismiss this view, but rather intends to complement it by arguing that, when it comes to considering computational systems, sensibility is not enough. Computation is a method of systematising reality through logico-mathematical means: an aesthetics of the computational should account for this specificity. But there is more. An aesthetics of the computational, in my view, should also acknowledge that indeterminacy in computation concerns the intelligible and the sensible alike. In this respect, my claim is as follows: there is an indeterminacy (or, as I

call it, a *contingency*) proper to the formal, logical, quantitative character of computation, and we need to take this indeterminacy into account when considering an aesthetics of computation in terms of a theory of how computation might determine itself.

My investigation of Gödel's and Turing's logico-mathematical discoveries is intended to present an initial step towards advancing such a case. I am aware that other histories of computation are possible, that the idea of a calculating machine has multiple origins, and that computational logic does not encompass all computing. However, this essay's focus on Gödel and Turing is a necessary beginning for any attempt to prove that an extension of computational aesthetics from the sensible to the intelligible is possible, and that it is possible precisely because indeterminacy is inscribed into the logic of computation. Whilst the focus of this essay rests upon the latter contention, the arguments presented within it form part of a broader project, which is intended to argue for this extension.

If we accept that *computation is a method of abstraction*, and that *aesthetics is a theory of experience*, then the aesthetic investigation of computation invites us to question the relation between what is abstracted and what is experienceable. It thus requires us to speculate whether this relation is even possible at all. The proposition that I advance in this essay calls for a full engagement with the formal abstractions of computation through a reworking of the limits and possibilities of *computational axiomatics*, which is the inferential structure that encapsulates these abstractions into a mechanism of automated inference. The commonly accepted conception of axiomatics is that it concerns a deductive structuring that is closed to change, variation and novelty, and which is thus fundamentally locked out from experience. *Contra* this widespread and popular assessment, I will use Gödel and Turing's discoveries to claim that the incomplete and the incomputable give us an 'open axiomatics', and that computation is marked by a contingent formalism. For the aesthetics of computation, I wish to argue, this proposition involves a reassessment of the potential of its formal structures. This is because the proposed ontology of the contingent computational structure allows us to think computational forms beyond the limits of formulae, and to engage with formalism beyond the idealisation of beautiful and truthful determinisms.

Of course the 'closure' and 'openness' in question are metaphorical properties. The history of computing is populated with closed or open constructs. A closed/open dichotomy involves, for instance, parallels with physical

instantiations of calculating machines (e.g. the openness or closure of an electric circuit), with information theory's transmission of communication over a channel, or with cybernetics' feedback models. The metaphorical closure that I address in this essay, however, is a more hypothetical and speculative attribute, denoting the conclusive character of transcendence and apriority that abstraction adopts in computational aesthetics, but also in computational culture at large. From this perspective, the abstract is expressed via logico-mathematical formalisation, which is assumed to bear no relation to the empirical if not for its unilateral and unidirectional imposition upon the perceptual and the physical.

Undeniably, computational systems are *formal axiomatic systems*. This means that they involve a starting point that is self-evident, and which defines the inferential steps that can be taken from it towards a result via the manipulation of a list of finite instructions. The traditional theory of computation (i.e. Turing's) is established and operates in parallel with the axiomatic nature of logico-mathematical abstraction. The latter is a way of reasoning that privileges deductive legitimacy over induction, and which holds that if premises are assumed to be true, then so too are the conclusions that one can deduce from them, with no need for empirical justification. The mechanical automation of the axiomatic method facilitates the possibility for computer science to be systematically closed. In a computer program and in an axiomatic system alike, everything is closed because everything is inferentially connected with everything else. Nothing is presented without being defined; nothing is affirmed without being proved. Axioms can indeed be considered to be computer programs.<sup>4</sup> Such conformity between the axiomatic and the computational can be established at many levels, but it is most clearly evidenced in their common deductive coherence. Non-contradiction is an absolute requirement for the axiomatic method and for computation. It is the consistency of self-evident propositions that locks, secures and, figuratively speaking, 'closes' the deductive systemic which axioms and computer programs both construct and depend upon.

Famously, the parallel between the axiomatic formalisation of a mechanical process and the deductive and discretising qualities of logical reasoning established on the basis of Turing's model of computation grew into the *computationalist paradigm*. This view holds computational states to be program descriptions of actual mental states, and cognitive activities to be comparable to the finite operations of computing machines. In the past decades, however, computationalism has had something of a bad press, most significantly due to

its implicit assumption that the rationalising calculations of formal axiomatic systems might be able to account for the entirety of the thinkable and the doable. At the intersection between computing, robotics and cognitive sciences, alternative models of intelligence and mentality have thus been proposed. These models articulate cognition as the relation between the inside of a mind (or of an abstract formal model of it) and the outside of an external factual world, and they attempt to make up for the supposed closure of Turing's computability by opening up the computational model to the environmental, the embodied, the empirical, the inductive.

A sort of general discouragement or distrust towards anything representational is echoed, culturally, in the intellectual belief that axiomatic operativity might be a vehicle for the enslavement of the world and of the human subject by the normative, exercised by means of instrumental rationality. Questions about axiomatic methodologies pertain, then, not only to logico-mathematical sciences, but also extend to views of society and culture, as well as to philosophical accounts of subjectivity, agency and reason. To conclude this point: axiomatics is very common today (one could say that it is in fact present at each touch of our fingertips upon the new computational devices that have proliferated within our societies), but it is not at all popular. The basic algorithmic (and therefore axiomatic) model of what computation is has been challenged also within fringe elements of computer science itself. New conceptualisations of the computing machine have been proposed, and attention has been directed not towards the deductive determination of logical inferences, but towards abstract models based on the *interaction* between worldly actualities, interpreted as the continuous directionality of composite parts. Similarly, 'unconventional', 'non-classical' and 'natural' computing are just some of the broad labels assigned to non-Turing or post-Turing explorations of computability, which attempt to open axiomatics to the biological and the physical realm, preferring the inductions of empirical sciences to the deductions of logic.

It has been necessary here to depict the status quo of axiomatics in computational culture in rather broad brushstrokes; what has been said should, however, suffice to contextualise the proposition that is advanced in this essay. Technically as well as philosophically, the problem with Turing's notion of computability is that its formalisation of 'valid reasoning' is entrapped within the golden prison of total determinism, according to which a finite and complete set of rules accounts for the full predictability of a mechanical system (such as Turing's discrete-state machine). From this perspective, computational

axiomatics can be accused of attempting to pre-programme thought and experience. This deterministic pre-programming of probabilities, I wish to argue, is indeed what the *metacomputational approach* amounts to, whenever it searches for an ordering technique to structure the diversities of life, and to thus master the whole organisation of the thinkable and the arguable. In this respect, the condemnation of axiomatics can be said to stem from a justified suspicion towards the metacomputational belief that rational calculation can explain and encompass every element of actuality. It is thus precisely from the critique of the ontological and epistemological consequences of metacomputation that one needs to start, in order to gradually build an argument for an open, and fundamentally non-metacomputational, reworking of axiomatics.

## Universal Computation

In Douglas Adams' comic sci-fi novel *The Hitchhiker's Guide to the Galaxy*, a race of hyper-intelligent, 'pan-dimensional' beings have devoted a lot of time and money into building a giant supercomputer named Deep Thought, in order to finally reveal the 'Ultimate Answer to the Ultimate Question of Life, The Universe, and Everything'.<sup>5</sup> In Adams' story, it takes Deep Thought seven and a half million years to run the programme, at the end of which the result is finally revealed: '42'. This number is the 'Ultimate Answer'. Unfortunately, the alien creatures quickly come to realise that the 'Ultimate Question' itself remains unknown.

Adams' famous joke provides a good fictional and parodic example of metacomputation: an approach that, in its broadest and most accessible articulation, calls for a comprehensive science of calculation through which one would be able to grasp the fundamentals of reality itself. This dream of a universal science of reasoning, which is classical in origin, is one of the main ideas of Western civilisation. One of its most relevant champions is the Baroque philosopher and polymath Gottfried Wilhelm Leibniz, who envisioned a complete discipline (called *mathesis universalis*) through which one would have been able to design a compendium of all human knowledge.<sup>6</sup> Modern attempts to automatise thought, proposed by mathematical logic, can be seen as being continuous with such rationalist experiments in 'metareasoning'. The lowest common denominator between these instances is found in the importance attributed to the employment of problem-solving strategies that, by specifying the rules of the game, expect that we can also know what we can achieve by playing it.

I will refer hereafter to the metacomputational approach as *Universal Computation*. The phrase is used for two reasons. The first is that it evokes the Universal Turing Machine and its general-purposiveness. Universal Computation, I would argue, pushes a metacomputational agenda, which in turn pursues a comprehensive science of 'valid reasoning'. According to this perspective, the concept of Universal Computation is strong and powerful, and succeeds in explaining: 1. how computing machines are supposed to work; 2. how such functionality implies a horizon of total operativity. In order to defend the latter claim, my second motivation for using the phrase 'Universal Computation' must now be introduced.

This reason concerns the specific onto-epistemological role that abstraction might be seen to assume in Universal Computation. It could be said that, following Turing, to compute means to quantify and to extrapolate, through a method, a finite rule that aims at the 'effective calculability'<sup>7</sup> of this quantification. The general-purposiveness of this epistemic endeavour makes claims of comprehensiveness plausible and viable. Universal Computation is, in this respect, the abstractive technique proper to deductive reasoning.

Metacomputational abstraction, however, is not only about knowing: *it is also about being*. Thus my second reason for employing the phrase 'Universal Computation' is that, by stressing the universality of the computational method, I can make the case for the onto-epistemological status of abstraction in computation. Drawing on the contention that epistemology constructs metaphysics (and vice versa), I believe that the functional determinism of Universal Computation also underpins a specifically metaphysical position. In other words, according to the metacomputational view, the epistemological techniques of deductive abstraction subtend ontological transcendence.

The abstract, in this sense, is a proper ontological status. In the metaphysics of Universal Computation, computational abstraction is above the empirical: software rises above hardware, and its formalism rises above its particular executable instantiations. Questions about metacomputational ontology are therefore questions about transcendent entities, i.e. entities that are not determined by others, which exist by virtue of logical necessity, and which are the foundation and principle of empirical things. Whilst, epistemologically speaking, the quality of metacomputational universality has thus originated from deduction, from an ontological view metacomputation is universal because it is based on transcendence. I believe that specifically Platonic positions could be seen to be the direct reference point here, insofar as they postulate a realist yet transcendent ontology of the universal. I am thus putting forth the view

according to which *mathesis universalis*, 'revamped' via twentieth-century attempts at formalising a definition of computability, is underpinned by a similar transcendent metaphysics to that of Platonic philosophy. In relation to the broader rationale of this essay, the question now becomes: what are the consequences of thinking Universal Computation as an onto-epistemological principle, according to which computation is held to be separate from empirical phenomena, and yet is also held to possess the power to account for what there is to know and be? And what does this tell us about its aesthetics?

## Beauty and Truth: Computational Idealism

"Beauty is truth, truth beauty,"—that is all Ye know on earth, and all ye need to know." It is probably safe to assert that when, in the second decade of the nineteenth century, the Romantic poet John Keats wrote these famous closing lines to 'Ode on a Grecian Urn',<sup>8</sup> he knew nothing of computation. His aim was perhaps to reflect on the relationship between art and life, and on the possibilities and constraints of the artistic medium. Equally, it can be supposed that when, in 2002, the free-software programmer and media artist Jaromil crafted a single-line piece of code that was able to bring down a whole Unix-like system, he was not paying homage to the neoclassical ideals celebrated by British Romanticism. Jaromil was less concerned with the orderly world of antiquity than with the very present and uncomfortable disarray of an operating system that crashes when a large amount of programs are run simultaneously, eating up all its memory and exhausting processor-time. His *ASCII Shell Fork Bomb* is a command-line script that does exactly this. By launching two or more copies of itself upon start up, the piece of code initiates a chain reaction in which the number of launches multiplies itself exponentially. The machine halts, incapable of coming to terms with the recursive spawning of processes. Rebooting is the only means of restoring normal functionality.

There is an undeniable historical, conceptual and programmatic distance between the two instances. The correspondence that I am suggesting is not meant to create any interpretative parallel; rather, Keats' statement can be viewed as an exemplification of the long-standing intellectual belief that beauty and truth are uniform and interdependent. This identity of beauty and truth echoes all the way from Pythagoras' numerology to the golden ratio of the Renaissance; it resonates in contemporary classes on computational logic and hits straight at the cursor in a terminal prompt, where the fork bomb is inputted. It is my contention that Universal Computation justifies and grounds, both ontologically and epistemologically, this classicist equivalence between

beauty and truth. The regulative and transcendent character of metacomputational abstraction thus enters computational culture and its aesthetics in terms of a commitment to a non-sensuous conception of the aesthetically worthy, and translates into classicist concerns such as beauty, elegance, harmony and simplicity. Logical necessity, on this view, becomes an aesthetic value, pertaining to closed, self-referential rules that are deductively true under all circumstances.

It is argued here that Universal Computation lies at the basis of an aesthetic approach to computation that I call *computational idealism*. The term 'idealism' is to be taken in its metaphysical sense, and indicates the principle according to which abstract laws are more fundamental to reality than that which we are presented with through perception and sensation. What I am depicting as an idealism of computation is a technocultural view that would maintain that computational structures are *ideal forms*, insofar as they are immutable entities, independent from matter as well as from contingency. From this perspective, aesthetic significance corresponds to a mathematical conception of beauty, articulated as the ultimate value, or as a cipher of computational being per se. The latter, in turn, is truthful insofar as it presents, through algorithmic inference, a means to an ideal of eternal and abstract formality that is essentially indifferent to change. What I have termed 'computational idealism' would thus seem to hold that there is equivalence between beauty and truth via *logical proof*: computational structures are truthful as they are logically consistent; this consistency is beautiful because it adheres to the axiomatic character of computation.

Like the Platonic solids of ancient tradition, beautiful and truthful computational structures are assumed to be permanent, closed and reliant only on their internal conditions of necessity. They are taken to be a priori, and to need no real-world evidence in order to exist. The ideal beauty of the closed system is evident in the example of Jaromil's fork bomb. Self-referential at most, the fork bomb draws a full circle from input to output and to input again, while showing no interest in flirting with the gaze of any beholder. While fork bombs and similar coding tricks are common in hacking culture, Florian Cramer comments that 'Jaromil manages to condense them to a most terse, poetic syntax, arguably the most elegant fork bomb ever written'.<sup>9</sup> The elegance that Cramer attributes to the script is achieved via syntactical density and through compression of notation and execution. From this point of view, Jaromil's piece complies with Donald Knuth's definition of elegance in computer science, which takes such elegance to be conditional upon the coder's proficiency in the 'art of

programming',<sup>10</sup> as much as upon quantitative concision and logical consistency.<sup>11</sup> Examples of the popularity and persistence of similar understandings of computational elegance can be found by looking at the plethora of programming manuals and textbooks that invoke the cleanliness (and therefore goodness) of 'beautiful code', as well as in those first-person accounts given by software developers who emphasise personal efforts towards minimising inscription whilst maximising functionality.

The reading of the aesthetic significance of the fork bomb that I propose here, however, intends to go beyond the appreciation of the digital mastery of the artist or the programmer. My aim is to focus on the fork bomb as a *closed formal system*, which is exactly the closed formal system presupposed, but also facilitated, by Universal Computation. I should at this point add that I am aware that Jaromil emphasises the fork bomb's status as an experiment on the cracks and gaps of the digital domain, and that he employs it to comment on the creative potential which he sees as intrinsic to the vulnerability of digital systems.<sup>12</sup> Thus, from Jaromil's perspective, the *ASCII Shell Fork Bomb* would seem to be less an inhabitant of the ivory towers of computational formalism than of the playgrounds of glitch and error. In a media art context, Jaromil's fork bomb can indeed be understood as a sort of computational ready-made: as something that explicates the auto-positing and self-affirming force of art; as something that stands between the recreational and the radical; as something that is a bit odd, but which, through artistic appropriation and popular misuse, is capable of reassigning the meanings and consequences of media systems.

Having clarified this, it should be stressed that, although I acknowledge these interpretations of Jaromil's work, I will not be addressing the fork bomb as a glitchy product of imperfect computation. This is because I am interested in viewing it in entirely the opposite manner; that is to say, in pursuing the sense in which Jaromil's script is neither an anomaly nor a bug, but rather a small program that does exactly what it is supposed to do. I believe that the work's exponential disruptiveness is not dependent upon the messiness of a machine running amok, but rather that the fork bomb can be disruptive purely because of the formal consistency of the script's logical formulation. The pace of the fork bomb is relentless, inexorable, inescapable. Marching towards the physical limits of the machine (i.e. memory and processing capacity), this piece of code is dangerous not just for the actual damages or the end-user discomfort that it can produce by halting the functionality of the operating system. In my opinion, the fork bomb seduces and scares as a result of the effectiveness of its method, and because of the intrinsic correctness of its proof. No less autarchic than

anarchic, Jaromil's *ASCII Shell Fork Bomb* does not care for external agents, and even less for the empirical to be part of its formula. Given the necessary conditions, it works independently from empirical phenomena as well as from users.

In this respect, the explicative and regulative force of computational idealism is appealing and almost irresistible: from a philosophical perspective, it is based on the presupposition that the intelligible dimension of eternal ideality might offer a means of regulating the phenomenal and the material, and that this normative character might be expressed through the transcendent nature of the realm of logico-mathematical formulation in respect to its sensible instantiations. This idealism of computational forms—and I take Jaromil's fork bomb to be example, however involuntary, of that idealism—might not be explicitly acknowledged in everyday computing practices. However, this concept is relevant and useful, because it can help to expose a certain way of looking at abstraction and experience in computation: one that is characterised by the transcendence of the abstract over the experienceable. Computational idealism, in this sense, might be seen to reiterate the Platonic top-down approach of the intelligible informing the sensible. My argument here is that this approach relies exactly on the onto-epistemological principle of Universal Computation that I have discussed above. The computational forms of computational idealism are transcendent insofar as they are independent from an empirical substratum, and crystallised in the prospect of Universal Computation. Metacomputation, then, not only epistemologically justifies a computational aesthetics of beautiful and truthful forms: it also ontologically underpins it.

Aesthetically, one can view this idealism of computational forms as lending itself to a focus on classicist aesthetic features (e.g. elegance, simplicity, symmetry, harmony) that purportedly ascend towards the unity and order of deductive abstractive methods. Epistemologically, computational idealism becomes a rationalist affair: it maintains deduction and inference as its fundamental mechanisms of invention and discovery, and endorses *mathesis universalis*' prospect of advancing, one day, an all-inclusive account of the workings of nature and mind. Moving from epistemology to ontology, computational idealism shares with metacomputational perspectives the belief that the abstract is metaphysically superior. Computational *logos* is intelligible, not sensible; it is thus metaphysical because it is indeed beyond or 'above the physical' (*metà tà physiká*) and has 'perseity' in virtue of the self-referentiality of its formal structures. Finally, from a cultural perspective, computation, when it is assumed to be both beautiful and truthful, comes to ground the all-too-

human faith in a computing machine that can do no wrong, as well as the technosocial construction of an image of infallible—and fundamentally impenetrable—algorithmic rationality.

## Ordering the Real: Forms become Formulae

Universal Computation, I claimed above, should be understood as a twentieth-century reformulation of *mathesis universalis*. This affirmation can now be developed further, by stressing that both Universal Computation and *mathesis universalis* on the one hand, and the aesthetics of computational idealism that is grounded on them on the other, rely on the axiomatic method for the automation and universalisation of procedures of 'valid reasoning'. It is upon the supposedly closed deductive circles of the axiomatic method that logical and mathematical structures are given an onto-epistemological primacy and an operative universality over sensibility. It is always through these 'closed' deductive circles that intelligible beauty enters into the realm of sensible fruition and, similarly, universal functionality into the dimension of effective application. A little more should thus be said about computation's formal axiomatic systems.

*Axiomatisation*, or the act of transforming a theory into a system of axioms, offers optimal deductive presentation, inasmuch as it eliminates references to content and does not resort to direct observation. The axiomatic method's disinterest towards the empirical is best attained through the employment of symbols, which encapsulate abstract forms into mechanisms of exact thought. *Symbolisation* is a distinctive feature of axiomatisation. It is also a central feature of computer programming, in which complex statements are reverted into higher principles and into an alphabet of symbols, a grammar and rules of inference. In computer science—just like in mathematics and logic—axiomatisation is thus conjoined to techniques of *formalisation*. Formal languages, rules of transformations and axioms are the defining elements of a formal system: elements that find one-to-one correspondence in digital computing machines. The isomorphism between the computational and the formal provides additional evidence of the closed character of deductive forms. Formalism itself, as a philosophical, historical and mathematical enterprise, is in the business of conclusion; it constructs conclusive systems through the combination and manipulation of expressions that are 'closed' because they are derived from the premises of the system itself.

Obviously, one must acknowledge the existence of many variations of formalism and of many different approaches to formal systems. To put it very generally,

formalism considers mathematics to be 'meaningless' and akin to a game of rule-governed syntactic manipulations, with no commitment to the reality of mathematical objects, regardless of whether the latter are considered to be residing outside the mind, or as cognitive constructs. It follows from this broad definition that there is an essential dissimilarity between strictly formalist ontologies and the Platonist metaphysics of a priori entities that computational idealism, I argued, would endorse. The former would imply a nominalist attitude; the latter, by contrast, a realist approach. While such an ontological contrast cannot be denied, it is worth remembering here that I am not interested in discussing formalism as a well-formed or illicit arrangement of characters. Equally, I will not focus on the notational development of formal languages in computation. Instead, I want to address formalisation in connection to the abstract conception of modern mathematics, according to which the discipline develops into a repository of *formulae*. Deductive abstraction becomes, then, *formulisation*: partly a tool to uncover the a priori principles of valid thought, and partly an instrument to enumerate these intelligible principles via the mechanisation of reasoning through symbols and formal languages.

With these issues in mind, I would now assert the following: that Universal Computation explains ontological and epistemological production precisely insofar as the latter responds only to the pure operational rule or, in other words, to the formula. From an epistemological view, metacomputational formulae are universal and transcendent functions of knowledge of the actual; from a metaphysical perspective, they convey being without origin and without change. Formulae describe calculations that are final, predictable and unavoidable. A formula can be thought of as a principle of proportion and measure from ancient tradition, according to which self-contained rules are 'cuts' towards a higher order of things. In its operative relevance, a formula is a recipe, a blueprint and a universally applicable procedure: a metacomputational organisation of the real via the instrumental paradigm into which formalism transforms abstraction.

According to an idealist perspective, empirical phenomena cannot be understood in themselves; one thus needs an abstract rule to make sense of them. If that is the case, then the role of the closed formula is to 'pre-form' the idea that will come to regulate the sensible. The primary relevance of axiomatic formulisation for a computational aesthetics of deductive forms is evident here. Computational idealism understands methods of abstraction according to axiomatic standards of reasoning. Ideal computational forms find in the formal

systems of computation a transcendent retreat from the empirical, as well as the possibility of being beautiful insofar as they are truthful (and vice versa). An aesthetics geared towards classicist questions such as beauty, elegance, simplicity, harmony is thus an aesthetics of the idealisation of a closed formulation of what can be intelligible.

## **A Way Out, but not Through the Empirical**

It has been necessary to discuss the metacomputational approach (Universal Computation) vis-à-vis its aesthetic counterpart (computational idealism) because I believe that the wished-for cause for the breakdown of the metacomputational paradigm has to be found in an inherent fallacy of the computational idealist approach. This fallacy concerns the non-relationality between abstraction and experience that lies at the core of computational idealism. According to computational idealism, a non-sensorial dimension of logico-mathematical intelligibility surpasses and yet regulates the empirical. It is my contention, however, that we should object to this transcendent schema, and that we should look at computational abstraction as something different, and something more, than metacomputational supremacy. My aim, in other words, is to 'liberate' computational forms from metacomputational formulation.

To that end, I wish to argue for a reconceptualisation of computational abstraction that would consider the indeterminacy of computation's formal axiomatic systems. I am thus arguing here for an understanding of computational abstraction that would allow us to think the contingency of computational systems. To some extent, my speculative hypothesis implies claiming that computation has its own 'experienceable' dimension. This is not the user's experience, nor the experience of an environment or of a situation. This is instead the experience (that is, as mentioned already, the self-determination) of the computational process itself. It is thus important to stress that I am not giving to computational formalism a phenomenal status. Rather, I want to find in computational formalism an *eventuality* that would permit me to address computational structures in their *actuality*.

This speculative operation could be attempted by reassessing the significance of Gödel and Turing's breakthroughs in relation to both Universal Computation and computational idealism. That the abstract formalisms of computation should be discarded in favour of phenomenal engagements is a conclusion that has often been advocated in various arenas of the study of computational media. I am

nonetheless contending here that Gödel's incompleteness and Turing's incomputability can be used as conceptual tools to argue for the opposite position: *that we should fully engage with formal abstraction and with the rational, discrete and axiomatic character of computation.*

Arguably, the understanding of a computational process as being a closed system has been challenged by technoscientific attempts to reformulate what computation is and what computation does. From this perspective, it could be added that to give to computation an eventuality, or indeed a contingent status, would seem to involve opening up the 'black box' of the computing machine to the indeterminacies and umpteenth variables of the empirical world. One can comment that this is exactly what experiments in ubiquitous, embedded and physical computing attempt to do, by bringing to the fore the question of how computing can be a conversational situation rather than mere symbol processing. Likewise, some academic fringes of computer science have explicitly attempted the formulation of broader and more intuitive definitions of computation and computability; definitions that would be considered less in terms of lists of instructions than in terms of descriptions of situated behaviour. These attempts also echo what is happening in some strands of the cognitive and neurosciences, where algorithmic programmability has been partly rejected on the basis that axiomatic rules cannot engage with the environmental complexity that makes us (and the machines that want to simulate us) intelligent.

However, in respect to such efforts to 'make computation empirical', so to speak, my proposition is conceptually different. In my view, the fallacy of computational idealism concerns not the fact that this idealist approach relies on axiomatics in order to bypass empirical factors, but rather the fact that this idealist approach fails to recognise that such axiomatics is always limited; indeed, *indeterminate*. The main point of concern of my critique of metacomputation, and of its related idealist aesthetics, thus involves *differentiating between the contingent and the empirical*. Rather than opening up computation to its phenomenal, worldly outside, I want to 'let it loose' on its indeterminate and yet still very formal inside.

I thus wish to suggest a speculative hypothesis. Our eyes are staring firmly at the axiomatic, and at the formal character of computation. I am asking the reader to take computational systems for what they are: formal axiomatic structures. However, I am also asking the reader to think about computational axiomatics in a different manner, and through a different understanding of

contingency qua indeterminacy in computation: *an understanding that would articulate the latter less in its empirical dimension than in its formal one.*

Computational axiomatics, I claim, is a body of knowledge that is not so self-contained; there is more to it than just a finite set of pre-determined instructions. This is my proposed strategy: the notions of incompleteness and incomputability are employed here not to impair axiomatic formal systems altogether, but rather to enhance the possibility of an 'open-ended' understanding of them. The central point of this approach is the assertion that incompleteness and incomputability call for another level, or plane, of the investigation of axiomatics; a level of enquiry that, by considering the non-metacomputational reality of abstraction, has the force to uncover that the limits of formal reasoning are not the limits of computational abstraction.

## **The Autonomy of Formal Axiomatic Systems**

The first step towards a critique of computational idealism and, consequently, of the metacomputational principle of Universal Computation that provides its onto-epistemological underpinning, involves bringing the debate about the limits of formal axiomatic systems to an implicit, yet extreme consequence: *the affirmation of the autonomy of formal axiomatic systems from the empirical.* Let us look at this question more closely.

Gödel's incompleteness theorems have drawn a great deal of attention from mathematicians and nonmathematicians alike, together with the consequential uses and misuses that such popularity often entails. According to one of the most common misconceptions, Gödel would be a sort of postmodern champion of relativism. From this postmodern view, his work on the limits of formal reasoning has helped to invalidate scientific objectivity, whilst also moving beyond quests for certainty and signification. Yet, in my opinion, the congeniality between Gödel's incompleteness theorems and postmodern relativism is only apparent, and perhaps as much the result of a misinterpretation of Gödel's mathematical work as of its profoundly philosophical connotations. Postmodern cultural theorists who hope to use incompleteness as the final nail in the coffin of metaphysics should thus be careful. From the perspective of his philosophy of mathematics, Gödel was a die-hard Platonist. This means that he believed in the objective reality of mathematical entities and concepts. His work on the foundation of mathematics was committed to a rationalist ontology of unchangeable and mind-independent mathematical entities, and was permeated by the desire to expand such 'conceptual realism' from mathematics to metaphysics. Saying that Gödel's

incompleteness theorems commend the dismissal of reason, then, means twisting the actual technical results of Gödel's 1931 proof, as well as overlooking his entire anti-empiricist metaphysical *weltanschauung*.

Turing's philosophical position, on the other hand, is much more implicit; it is thus difficult to reconstruct it from the scarce body of publications that he left upon his premature death. His biographer, Andrew Hodges, comments that the seminal 1936 paper 'On Computable Numbers' appeals to 'operations that could actually be done by real things or people in the physical world'.<sup>13</sup> In other words, the paper addresses the concreteness of calculative procedures. This is surely true. However, I believe that it is possible to argue that computation, via Turing's groundbreaking work, is formalised into a process of producing and collecting abstracted entities, and of showing how a conclusion necessarily follows from a set of premises. This computational activity of pinning down chains of abstract reasoning into a fixed formal structure can thus be seen as something that has to do with the necessary and the a priori. It upholds the demand for the self-evidence of first principles, albeit (and this is a crucial distinction) within the constraints, limits and confinements of the system in question. In this sense, one could read Turing's 'effective method' of calculation as an exploration of the possibility of predicating deductive thought uniquely on its extra-empirical terms. To conclude this point: my contention is that both Gödel and Turing never intended to abandon the ship of deductive reason, or sought to imply that that ship was sinking. Incompleteness and incomputability do not suggest an arbitrary 'anything goes', or that deductive abstraction is inescapably mistaken. On the contrary, they prove that logico-mathematical reasoning cannot be contained within a finite formulation.

The impossibility of enclosing logico-mathematical procedures into any final systematisation is one of the most important consequences that can be drawn from Gödel's incompleteness theorems and Turing's account of incomputability. Yet, although this impossibility has been traditionally read as evidence of the failure of axiomatic systems, I would contend here that Gödel and Turing's breakthroughs attest that axiomatics does not correspond to empirical individuations, with direct consequences for the metacomputational principle of Universal Computation on the one hand, and for the aesthetics of computational idealism on the other. Hilbert hoped for a twentieth century where 'there is no *ignorabimus*'.<sup>14</sup> It is possible to argue that his metamathematical initiative thus moved within the assumption of metacomputation. That is to say, it accorded with the conviction that a complete circle of logical necessity can be drawn from premises to conclusions, and that it is possible to fulfil this circle in a finite

number of steps. The spectacular demise of Hilbert's programme, as caused by the logical genius of Gödel and Turing, showed however that these steps are not finite, and that the inferential circle cannot be closed. This discovery defeats any attempt to pursue a comprehensively valid inferential science of reasoning. Moreover, I wish to assert, it also entails the independence of axiomatics from the metacomputational project that would attempt such a pursuit.

This independence is, in my view, one of the most urgent effects of the incomplete and the incomputable for computational aesthetics. Incompleteness and incomputability both show that computational forms are not metacomputational formulae. The axiomatic method is supposed to frame such forms into strictly deductive and reductive 'simple' formalisations. However, incompleteness and incomputability demonstrate that such a method cannot ultimately account for the transcendent relation that Universal Computation had aimed to authenticate by appealing to deductive operations that would close the circle of their ideation and actuation via simplification. Ultimately, Universal Computation falls short of providing an entirely workable account of the exactness and certitude of its axioms because the transcendent ontology to which it appeals fails to map a completely deductive path from abstracted objects to concrete entities. The equivalence of beauty and truth that grounds the computational idealism of certain computational practices is predicated instead on the possibility of walking that regulative path from start to finish. A closed formal systemic, I argued, is what grounds computational idealism's equivalence between logical truth and aesthetic beauty. That this inferential route is less a closed circle than a 'strange loop', to paraphrase an expression of Hofstadter,<sup>15</sup> is something that Gödel and Turing demonstrated more than eighty years ago. This demonstration, I believe, does not amount to saying that beauty and truth do not exist. Rather, it shows that there cannot be a 'golden ratio' that will provide us with the exact proportion of their relation. Via Gödel and Turing, we understand that there cannot be a complete deduction of everything: axioms cannot even explain themselves.

Rebecca Goldstein emphasises that 'according to Gödel's own Platonist understanding of his proof, [its paradoxical nature] shows us that our minds, in knowing mathematics, are escaping the limitations of man-made systems, grasping the independent truth of abstract reality'.<sup>16</sup> Following Goldstein's assertion, one can ask: what does this 'independent truth of abstract reality' involve? In my opinion, the answer to that question is that such a reality is beyond the plane of metacomputational organisation and representation. Perhaps most telling of all is the fact that both Gödel's proofs of incompleteness

and Turing's computational limits can be understood to have rephrased the impossibility for man-made formal encodings to exactly mirror the empirical world. On the one hand, the instrumental prospect of a final axiomatic systematisation cannot be matched by any particular concrete instance. On the other hand, the transcendent relation between the purely intelligible and the given, manifest and sensible reality is also proved to be largely impractical, if not biased, when attempted via formal symbolic reductions. This mismatch between the ideal and the empirical shatters the fundamental premise of *mathesis universalis*, i.e. the assumption that a transcendent ontology of ideal formality can attain a conclusive systematisation of the real by means of deductive calculation.

## Open-Ended Axiomatics

It is well-known that Gödel advocated rational intuition as affording privileged access to mathematical certainty, stressing that his work in mathematical logic does not determine 'any bounds for the powers of human reason, but rather for the potentiality of pure formalism in mathematics'.<sup>17</sup> The axiomatic way is, for Gödel, still the paradigmatic way towards valid reasoning. In his opinion, however, only the human faculty of mathematical insight can lead us along that road, by way of a mixture of intuition and deduction. Axioms, Gödel wrote, 'force themselves upon one as being true'.<sup>18</sup> Mathematical reasoning partakes of conceptual imagination and ingenuity, and cannot possibly be limited to a rule-following activity. *Contra* formalist games of symbolic manipulation, Gödel urged that the task of securing the certainty of mathematics could be performed 'by cultivating (deepening) knowledge of the abstract concepts themselves which lead to the setting up of these mechanical systems [...]'.<sup>19</sup> Gödel's rational intuition is not a special kind of sense perception, but rather its 'a priori analogue'.<sup>20</sup> It is 'something like a perception',<sup>21</sup> which can access and accommodate even those abstract entities that show a certain 'remoteness from sense experience' (ivi). While, for Hilbert, intuition had to appeal to concrete things (and, according to its formalist proposition, it could yet be replaced by finite sign manipulations), for Gödel intuition was the 'sudden illumination'<sup>22</sup> through which one grasps the essence, the form or the idea of abstract mathematical objects.

In Gödel's rational intuition the perceived object might not be an empirical phenomenon, and might in fact be an Idea, or the essence of a mathematical entity; it is possible to argue, nonetheless, that this object remains phenomenal insofar it is given to the consciousness of a subject that is already pre-defined.

This is a tricky situation: figuratively speaking, while metacomputational transcendence left the stage through the main door, a sort of transcendental subject would now seem to make its arrival from a side entrance. The refined intricacies of Gödelian epistemology are not an explicit concern of this essay. It suffices here to stress that Gödel's intuition can be understood as a rational, yet still existential insight into the logico-mathematical a priori; an a priori that is given to human intuition as a kind of non-sensory datum. There is a gap between the ideal and the empirical. This fracture has scared off man-made symbolic mechanisms, showing them to be superfluous and always incapable of accounting for the inexhaustibility of logico-mathematical thought.

Nevertheless, according to the Gödelian epistemological framework, a metaphorical 'eye of the mind' can gaze past this metaphysical breach. In some cases, it can even manage to contemplate what is beyond that rupture. This intuitive sight is always subjectively experiential. Gödel's open-ended orientation of the axiomatic method, therefore, takes place from what we could call a 'subjectivist' view. Formal rules and procedures cannot justify axiomatics; only this transcendental point of view can, through glimpsing the reality of mathematical objects.

It follows from these comments that, whilst Gödel's incompleteness has helped me to argue for the autonomy of formal axiomatic systems (and it is indeed key to that argument), Gödel's very own philosophical endorsement of rational intuition can only bring me up to a certain point in the elaboration of such autonomy vis-à-vis automated systems. Consequently, I will hereafter focus on Turing's incomputability, for the incomputable adds an important non-existential 'flavour' to the open-endedness of axiomatics that I wish to pursue. Gödel's axiomatics gives itself to human intuition as an experientially personal construction. Conversely, Turing's axiomatics opens inwardly and impersonally, towards itself. This condition requires me to propose a profound and radical reassessment of the significance of the limits of computation's formal axiomatic procedures.

Simply put, the claim to be advanced at this point is this: incomputability allows me to look into the limits of formalism that Gödel envisaged, and to do so vis-à-vis its utmost mechanisation, i.e. via the computational automation of formal procedures. Axiomatics might as well be accessible to human reason. What I wish to stress, nonetheless, is that *formal axiomatic systems are exposed to their own indeterminacy, before any empirically-based attempt to access them*. Turing's mechanical formalisation of deductive abstraction, I would argue, proves that rules have a level of contingency that does not depend on the

empirical outside of the formal axiomatic system. But how does Turing's incomputability point towards this direction? The answer, I would argue, is through the infinite.

## **The Discretisation of Infinity**

At the core of Turing's investigation of computability there is a transversally posed, but crucially metaphysical problem: how is it possible to define the infinite in finite terms? The infinity in question is that of real numbers: a class of values representing quantities along a continuum, like points in a line, whose digits are infinite and therefore uncountable. The finite steps, conversely, are those of the deterministic method used by Turing to cope with the dilemma itself. Modelling the experiment on what he hypothesised to be the human mind at work, Turing divided the machine's functionality into small operations, whose purpose and conclusion were ruled by their own configuration, and whose actions were set out by an equally finite table of behaviour. A Turing machine program is thus an effective method of calculability facing the possibility of encountering real numbers that cannot be computed within any desired certainty or precision via an enumerable (and hence countable) set of instructions. The impossibility of processing the vast majority of real numbers through a mechanical procedure conceptually couples infinity and decidability in a binding logical argument: it is impossible for any Turing machine to prove if any other similar device would be able to write (which means, to calculate) the infinite digits of a real number on its blank tape.

At first, this conclusion would only seem to be relevant within mathematical scholarship. However, it is worth noting that it is because of incomputability that, to this day, there does not exist a one hundred per cent 'bug-free' computer program. The correctness of a piece of software cannot be finally determined because of the inherent limits of algorithmic methods of testing, which check a computer program by executing the tested code in a finite-step, line-by-line manner. The infinite possibilities of endless loops in a program are beyond the finiteness of the time-constrained procedure constructed to infer its fallacy. We just do not know until we execute or run the program itself. We just do not know, for instance, if a program will halt unless we run such a program, and we cannot tell which inputs will lead to an output. This condition can also be said to pertain to the reasons for the contemporary software industry's tendency to neglect its capacity for generalisation in favour of more specific, limited solutions, viable only within certain contexts and parameters. In concrete terms, rather than engaging in creating software that provides a

universal answer for a certain requirement, the industry finds it more relevant to provide the market with very specific products, whose finite, algorithmic determinacy is thus easier to be tested. Constraints, restrictions, bounds and iterations are added to the software, along with the pseudo-consolatory interface option for the end user to terminate a process that shows no hope of halting.

These and equivalent programming practices are useful examples vis-à-vis the conceptual point that I am making here. They appear to relegate the relevance of mathematical infinity to the very peripheral role of an inconvenience: something of a theoretical annoyance in the solution of a problem via 'closed' axiomatic means, and thus a distraction to be estranged from the counting of zeros and ones. Computer science is often quite pragmatically indifferent to 'Cantor's paradise' (namely, Georg Cantor's thesis about the existence of actual infinity, hailed by David Hilbert himself as the final conceptual *locus* of the whole science of mathematics). Such a 'heavenly' status is not a shared destiny for the numbers crunched in and out by computing machines. Computational 'virtues'—to keep the eschatological metaphor—are rather defined in terms of the capacity to break down infinitely many data into enumerable finite sets of processes, eliding anything that will not fit a univocal correspondence between proof and execution.

Of course this agnosticism in computability theory is so much more than mere *horror infiniti*. The questions that it raises cannot be dismissed as a clever tactic to avoid the headaches of mathematical counting. From this perspective, it can be said that computing machines are constructed upon the twentieth-century discovery of the logical deadlock between finitude and infinity. The issue that I would now like to address is this: what happened to formalism once it has been proved that there are limits beyond which the deduction of abstract systems necessarily confronts (and even generates) undecidability? Furthermore, how is the axiomatic method complicated and challenged by incomputability, and yet equally envisaged as the artificial and fully automated process that stands at the very cusp of the domain of instrumental rationality?

My hypothesis is as follows: Turing's theory of computation appears to be grounded on formalisation understood as a method of *discretisation*. One could object that many, if not the majority of formalising systems of thought can be seen to confront the question of how to relate the discreteness of their symbols and rules with the sensible continuity of the empirical reality to be encrypted. While I agree with this objection, my own point here as regards computational

formalism is slightly different. It pertains to computation's validation of the discretising formalism of its systems, via the logical discovery that rules of deduction are limited, and yet useful all the same. These limitations, and this usefulness, are both established through the conceptual and mechanical discretisation of logico-mathematical infinity. Computer science (and Turing himself) might have pragmatically adopted a posture of seeming indifference towards quantitative infinity. Nonetheless, my contention is that this quantitative infinity in computation should be considered the motor of Turing's model for the computational mechanisation of deductive abstraction. At the core of Turing's model of computation there is a quite revolutionary proposition: computation is made up of quantities, yet quantities that cannot be fully counted. With Turing's incomputability one thus witnesses something especially surprising: it is the mechanical rule itself that, in its operations of discretisation, generates the inexhaustibility of computational processing.

Incomputability, I wish to claim, brings Gödel's results beyond the thought of Gödel himself. Of course, the question of infinity was already implicit in Gödel's incompleteness, according to which (and here I am simplifying Gödel's proof a great deal) more and more axioms can be added to the formal system. With incomputability, however, not only is the very existence of axiomatics freed from a metacomputational systematisation, but so too is the regulative and procedural character of the axiomatic method. This latter becomes, in principle and in effect, non-metacomputational, because it is opened up to infinity. Turing's notion of computability is founded upon a quantification of logico-mathematical infinity. Such quantification, however, cannot be fenced within a horizon of totality, for between input and output there is an infinity of quantities. These quantities, however, are not metacomputational measures and amounts, but rather the quantitative character of a function that is determinable and yet not decidable, and which is not, therefore, computable.

The limits of computation, then, should not be understood as conclusive restrictions or constraints, but in fact as an aperture towards quantitative infinity. From this standpoint, it can now be argued that computational forms are *not* metacomputational formulae. In other words, incomputability means trouble for a 'golden ratio' of beautiful and truthful ideal forms. This is because incomputability shows that such forms in computation are infinite quantities that cannot be composed into a closed totality, upon which the equivalence between logical validity and aesthetic value was established. Consequently, the formal axiomatic systems of computation are not exhausted as sets of pre-programmed rules. The pre-programmed axiomatic rule itself contains an

internal level of undecidability that cannot be systematised by rationing, measuring and counting, despite the fact that it is exactly these actual procedures of systematisation that engendered the very possibility of such undecidability.

However, an important qualification needs to be added: computational formalisms work. In fact, they work very well. The foundational crisis over the superstructure of mathematics might have permanently undermined logicism by demonstrating that there are consequences that cannot be said to necessarily follow from initial assumptions. At the same time, though, the relatively silent establishment of a theory of computation urgently re-problematized the relation between the ideal and the empirical via the paradox that there are things that can be described finitely but that cannot be computed. To put it otherwise: whilst doubts about rigorous criteria for a satisfactory foundation of mathematics have profoundly threatened the possibility of reducing mathematics to logic, Hilbert's critical proposal for a 'word-game' method of formalisation is kept alive and well by the investigation of the computational relation between rules for deductive abstraction and calculability. In the shadow of the failure of Hilbert's quest for metamathematical proofs, it is his formalist study of mathematics as a mechanical and context-independent symbolic manipulation that seems to have found, in the theory of computation, both its own declaration of intents and the examination of its possibilities and limitations.

## **Contingency and the Event of Computation**

To some extent, Turing was looking for the metacomputational formula. He actually managed to offer a model for a general-purpose, discrete-state machine by which to attempt universal calculations. However, it should also be stressed that the greatness of Turing's 1936 work lies in seeing that the comprehensive character and regulative scope that Universal Computation would seem to ascribe to deductive general-purposiveness is not attainable. Thus, whilst on the one hand Turing's incomputability confirms that Universal Computation works in terms of the operability of an effective method of calculation, on the other it also definitely disproves the transcendent premises on which the metacomputational approach is established and developed.

In this respect, my contention is that this overthrowing of transcendence does not occur through the dismissal of computational formalisms, but through the affirmation that these formalisms have in fact too much reality. *This is the*

*reality of their limits*. By linking this point to my previous claims concerning the infinite quantities of computation, one can now add that to understand formalisation as a process discretisation, and to do so vis-à-vis Turing's incomputable, means to recognise that computation is 'limited' by its being open towards its own infinity. Here the word 'limit' is somehow misleading, because it would seem to imply a shortcoming of sorts. What I want to propose, instead, is the contrary view: the limits of computation are the mark of computation's *potentiality*. The fact that, within a computational formal axiomatic system, there are things that cannot and will not be calculated opens up the possibility for computation to be something different than a *mathesis universalis*, and for the formal axiomatic structures, which computation employs, to be less self-contained than such a *mathesis universalis* would assume them to be.

From this perspective, I believe that to understand the limits of computation as computation's potential involves retheorising the quantities of digital computational systems. The potentiality of computation that one finds via Turing's incomputability is not a qualitative or continuous analogical field of transformation, but the discrete and intrinsically computational perspective that quantities might or might not be determined in the calculation. According to a metacomputational approach, quantities compose a closed pattern of logical necessity, which would consequently give rise to aesthetic worth. Yet, what Turing demonstrates is that these quantities might not necessarily produce an output from their deductive operations by virtue of their quantitative infinity. This discovery is paradoxical, because it makes the event of incomputability a sort of internal 'sufficient reason' of computational functions.

The expression 'sufficient reason' is surely very philosophically charged, but perhaps it can be used here speculatively, so as to convey the idea that there is nothing very irrational in incomputability; incomputability, on the contrary, is the reason of every computation. The rationality of computing, then, extends well beyond the rationalistic means of Universal Computation, in a way that metacomputation cannot predict. This rationality, in other words, is not the Leibnizian total determinism of an a priori that calculates, transcendently, the reason of things, in a metacomputational fashion. Rather, sufficient reason here is the Whiteheadian ontological principle according to which events are all there is in reality. This rationality is thus the *self-sufficiency* of computation itself. Counterintuitively, the incomputable is an excess of reason, not the lack thereof.

In my view, then, the most interesting result of formalisation-as-discretisation lies not in the fact that the qualities of experience cannot be fully encapsulated in the computational function, but in the fact that this function, which is quantitative in nature, has its own way of being profoundly indeterminate. In other words, *it is contingent*. Contingency is thus conceptually separated here from empirical qualities as well as from an existential or personal point of view. The contingency of the computational event is not the contingency of the phenomenal or of the sensible, nor the contingency of a transcendental subject. With the incomputable, contingency becomes the condition of the infinite quantities of computation; a condition that is nevertheless inherent to the formal axiomatic systems of computation. Of course, Turing never says this in his 1936 paper. The aim, scope and context of his logical and mathematical enquiry were very different. Nevertheless, it has been claimed here that an argument for the contingency of formal axiomatic structures is one of the most striking conceptual consequences that one can develop speculatively from the results of Turing's work on computability. The speculative proposition that I would urge us to consider is this: *there is a contingent ontology of computation that is to be found and discussed at the formal level*.

The argument that I am proposing is thus, fundamentally, an argument for the *autonomy of computation*. This implies a realist stance: computation is neither about mathematical idealism nor physical empiricism; computation is computation. Of course, it is not my intention to depict a world where there are no uses of the computational. Undeniably, computation is part of what constructs society, economy and culture today. Perhaps, however, we can understand computation's role in this construction better if we allow for the study of computation's very own and very specific contingent ontology. To this end, computational formalism is not to be hailed as the measure of the universe, but neither is it to be discarded because it is incapable of fully encompassing the confusion of the real world, or the behavioural richness of living and lived experience. The point is, strictly speaking, that computation does not necessarily pertain to such an all-encompassing view; for before it sets about addressing the contingency of the world, computation ought first to account for its very own indeterminacy. Via Turing's incomputability, it is possible to argue that rule-following is not as straightforward as it was thought to be. In the autonomy of the procedure, in its rational mechanics, something unpredictable is generated. The specificity of computation is that at the heart of its formalism there is a contingent event, which is the determination, vis-à-vis indeterminacy, of the computational process itself. So, whilst it has been suggested that we should abandon formalism because it brings us to the verge

of transcendence, I would propose the opposite: that we should engage with it because the immanence of formalism's contingency dwells at its heart.

It is here that one needs to return to the claim that the study of the contingent ontology at the heart of computational formalism is, primarily, an *aesthetic investigation*. For computational aesthetics, the contingent ontology of computation is a radical hypothesis, inasmuch as it changes the terms of the relation between abstraction and experience in computation. At the outset of this essay I have argued that any aesthetic enquiry of and into computation begs the question of the relation between abstraction and experience in computational systems. That there cannot be such a relation because of the transcendence of computational forms is the position of what has been referred to here as computational idealism. It can also be speculated that perspectives that want to embody, embed or situate computation could be said to want a relation of such kind, but that for them this relation has to pass via a reciprocal determination with sensibility, and a physical transduction thereof. From the standpoint of an *incomputable aesthetics*, however, the relation is already immanently established with every computation, because the formal structure has its own way of being eventual, and therefore of 'experiencing' insofar as it determines itself vis-à-vis indeterminacy. Via the notions of incompleteness and incomputability one discovers that the inherent indetermination of computation pertains to its intelligible dimension, and that this indetermination is encountered by way of abstraction: this is a formal indeterminacy, not an empirical one. These considerations, ultimately, open up the possibility to propose an aesthetic ontology of computational procedures in their logico-quantitative specificity.

I am using the expression 'incomputable aesthetics' here in order to link the aesthetic investigation of computation with the ontological argument that I have drawn from incomputability. However, another reason for employing the expression is the fact that this aesthetic ontology is indeed not computable from the point of view of any metacomputational aim. An *incomputable aesthetics*, therefore, is part of the conceptual legacy of the discoveries of Gödel and Turing. These discoveries energise the philosophical investigation of what aesthetics in computation might be, insofar as they might be said to legitimise the study of formal axiomatic structures beyond any classicist or idealist concern. One can indeed talk of an aesthetics of computational formalism without transcendence, but with a renewed attention for abstraction. This is, I believe, the greatest contribution that the incomplete and incomputable give to computational aesthetics: we are given the possibility of thinking again what

formal abstraction is, ontologically, and of thus confronting the speculative richness of the formal axiomatic structure. The legitimacy of this aesthetic study does not come from mathematics or from logic. It comes from computation itself. The aesthetics of contingent computation is not the only computational aesthetics that is possible, although it is perhaps a key one, as it strikes at the core of what computation is and does, before the human, but also beyond the post-human. This is, in other words, the aesthetics of the computational function itself: one that is not in opposition to digital aesthetics, new media aesthetics, user aesthetics, web aesthetics, interface aesthetics, interactive aesthetics and all the other aesthetic studies of the computational, but one that might be conceptually capable of ontologically orienting them.

## Biography

M. Beatrice Fazi is Research Fellow in Digital Humanities & Computational Culture at the Sussex Humanities Lab (University of Sussex, United Kingdom). Her research explores questions at the intersection of philosophy, science and technology. Her current work investigates the limits of formal reasoning in relation to computation, and it addresses the ways in which these limits shape the ontology of computational aesthetics. Through conducting this work, she aims to offer a reconceptualisation of contingency within formal axiomatic systems vis-à-vis technoscientific notions of incompleteness and incomputability.

## Bibliography

Adams, Douglas. *The Hitchhiker's Guide to the Galaxy*. London: Pan Books, 1979.

Chaitin, Gregory. *Meta Maths. The Quest for Omega*. London: Atlantic Books, 2005.

Cramer, Florian. *Entering the Machine and Leaving It Again: Poetics of Software in Contemporary Art*. 2006. Accessed 11 February 2014.  
[http://gwei.org/pages/press/press/Florian\\_Cramer/fullversion.html](http://gwei.org/pages/press/press/Florian_Cramer/fullversion.html).

Gödel, Kurt. "On Formally Undecidable Propositions of the Principia Mathematica and Related Systems I." In *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvability Problems and Computable Functions*, edited by M. Davis, translated by E. Mendelson. 4-38. Mineola, NY: Dover Publications, 2004.

Gödel, Kurt. "On Undecidable Propositions of Formal Mathematical Systems." In *Collected Works. Volume I: Publications 1929-1936*, edited by S. Feferman, J. W. Dawson Jr., S. C. Kleene, G. H. Moore, R. Solovay, and J. van Heijenoort, 346-372. Oxford: Oxford University Press, 1986.

Gödel, Kurt. "The Modern Development of the Foundations of Mathematics in the Light of Philosophy." In *Collected Works. Volume III: Unpublished Essays and Lectures*, edited by S. Feferman, J. W. Dawson Jr., W. Goldfarb, C. Parsons, and R. N. Solovay, 374-387. Oxford: Oxford University Press, 1995.

Gödel, Kurt. "What is Cantor's Continuum Problem." In *Philosophy of Mathematics. Selected Readings. Second Edition*, edited by P. Benacerraf and H. Putnam, 470-485. Cambridge: Cambridge University Press, 1983.

Goldstein, Rebecca. *Incompleteness. The Proof and Paradox of Kurt Gödel*. New York: W. W. Norton & Company, 2006.

Hilbert, David. "Mathematical Problems. Lecture Delivered Before the International Congress of Mathematicians at Paris in 1900." *Bulletin of American Mathematical Society* 8 (1902): 437-479.

Hodges, Andrew. *Alan Turing. A Short Biography. Part 3: The Turing Machine*. 2005. Accessed 11 February 2014. <http://www.turing.org.uk/bio/part3.html>.

Hofstadter, Douglas. *Gödel, Escher, Bach. An Eternal Golden Braid*. London: Penguin, 2000.

Jaromil. `:( ) { : | : & } ; : - ou de la bohème digitale`. 2002. Accessed 11 February 2014. <http://jaromil.dyne.org/journal/forkbomb.pdf>.

Jaromil. *ASCII Shell Forkbomb*. 2002. Accessed 11 February 2014. [http://jaromil.dyne.org/journal/forkbomb\\_art.pdf](http://jaromil.dyne.org/journal/forkbomb_art.pdf).

Keats, John. *Complete Poems*. Cambridge, MA: Belknap Press, 1982.

Knuth, Donald. *Literate Programming*. Stanford, CA: Center for the Study of Language and Information, 1992.

Knuth, Donald. *The Art of Programming. Vol. 1: Fundamental Algorithms*. Upper Saddle River, NJ: Addison-Wesley, 1997.

Leibniz, Gottfried Wilhelm. "Preface to a Universal Characteristic (1678-79)." In *Philosophical Essays*. Translated by R. Ariew and D. Garber, 5-10. Indianapolis, IN: Hackett Publishing Co., 1989.

Turing, Alan M. "On Computable Numbers, with an Application to the Entscheidungsproblem." *Proceedings of the London Mathematical Society* 42 (1936): 230-265.

Wang, Hao. *A Logical Journey. From Gödel to Philosophy*. Cambridge, MA: The MIT Press, 1996.

## Notes

1. Alan M. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem," *Proceedings of the London Mathematical Society* 42 (1936): 230-265. ↵
2. See Kurt Gödel, "On Formally Undecidable Propositions of the Principia Mathematica and Related Systems I," in *The Undecidable: Basic Papers on Undecidable Propositions, Unsolvability Problems and Computable Functions*, ed. M. Davis, trans. E. Mendelson (Mineola, NY: Dover Publications, 2004), 4-38. ↵
3. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem", 249. ↵
4. Gregory Chaitin, *Meta Maths. The Quest for Omega* (London: Atlantic Books, 2005), 65. ↵
5. Douglas Adams, *The Hitchhiker's Guide to the Galaxy* (London: Pan Books, 1979), 130. ↵
6. Gottfried Wilhelm Leibniz, "Preface to a Universal Characteristic (1678-79)," in *Philosophical Essays*, trans. R. Ariew and D. Garber (Indianapolis, IN: Hackett Publishing Co., 1989), 5-10. ↵
7. Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem", 231. ↵
8. John Keats, *Complete Poems* (Cambridge, MA: Belknap Press, 1982), 282-283. ↵
9. Florian Cramer, *Entering the Machine and Leaving It Again: Poetics of Software in Contemporary Art*, 2006, accessed 11 February 2014, [http://gwei.org/pages/press/press/Florian\\_Cramer/fullversion.html](http://gwei.org/pages/press/press/Florian_Cramer/fullversion.html). ↵

10. Donald Knuth, *The Art of Programming. Vol. 1: Fundamental Algorithms* (Upper Saddle River, NJ: Addison-Wesley, 1997). ↵
11. See Donald Knuth, *Literate Programming* (Stanford, CA: Center for the Study of Language and Information, 1992). ↵
12. Jaromil, *:(){ :|:& };; – ou de la bohème digitale*, 2002, accessed 11 February 2014, <http://jaromil.dyne.org/journal/forkbomb.pdf>. See also Jaromil, *ASCII Shell Forkbomb*, 2002, accessed 11 February 2014, [http://jaromil.dyne.org/journal/forkbomb\\_art.pdf](http://jaromil.dyne.org/journal/forkbomb_art.pdf). ↵
13. Andrew Hodges, *Alan Turing. A Short Biography. Part 3: The Turing Machine*, 2005, accessed 11 February 2014, <http://www.turing.org.uk/bio/part3.html>. ↵
14. David Hilbert, "Mathematical Problems. Lecture Delivered Before the International Congress of Mathematicians at Paris in 1900," *Bulletin of American Mathematical Society* 8 (1902): 445. ↵
15. Douglas Hofstadter, *Gödel, Escher, Bach. An Eternal Golden Braid* (London: Penguin, 2000). ↵
16. Rebecca Goldstein, *Incompleteness. The Proof and Paradox of Kurt Gödel* (New York: W. W. Norton & Company, 2006), 50-51. ↵
17. Kurt Gödel, "On Undecidable Propositions of Formal Mathematical Systems," in *Collected Works. Volume I: Publications 1929-1936*, ed. S. Feferman, J. W. Dawson Jr., S. C. Kleene, G. H. Moore, R. Solovay, and J. van Heijenoort (Oxford: Oxford University Press, 1986), 370. ↵
18. Kurt Gödel, "What is Cantor's Continuum Problem," in *Philosophy of Mathematics. Selected Readings. Second Edition*, ed. P. Benacerraf and H. Putnam (Cambridge: Cambridge University Press, 1983), 484. ↵
19. Kurt Gödel, "The Modern Development of the Foundations of Mathematics in the Light of Philosophy," in *Collected Works. Volume III: Unpublished Essays and Lectures*, ed. S. Feferman, J. W. Dawson Jr., W. Goldfarb, C. Parsons, and R. N. Solovay (Oxford: Oxford University Press, 1995), 383. ↵
20. Goldstein, *Incompleteness. The Proof and Paradox of Kurt Gödel*, 122. ↵
21. Kurt Gödel, "What is Cantor's Continuum Problem," in *Philosophy of Mathematics. Selected Readings. Second Edition*, ed. P. Benacerraf and H. Putnam (Cambridge: Cambridge University Press, 1983), 483-484. ↵
22. Hao Wang *A Logical Journey. From Gödel to Philosophy* (Cambridge, MA: The MIT Press, 1996), 169. ↵