Preface: Weird Formalism

This book is about the logic of *computation*[1] and its ingression into culture. It describes a world in which *algorithms* are no longer or are not simply instructions to be performed, but have become performing entities: actualities that select, evaluate, transform, and produce data. In this world, algorithms construct the digital spatiotemporalities that program architectural forms and urban infrastructures, and are thereby modes of living. This is not to contend that algorithms are the building blocks of a physical universe in which any kind of thought can be fully computed. Instead, a closer look at algorithmic procedures shows that incompleteness in *axiomatics* is at the core of computation. These performing entities—algorithms—expose the internal inconsistencies of the rational system of governance, inconsistencies that correspond to the proliferation of increasingly random *data* within it. Instead of granting the infallible execution of automated order and control, the entropic tendency of data to increase in size, and thus to become random, drives infinite amounts of *information* to interfere with and to reprogram algorithmic procedures. These entropic bursts of data within computation add new information to the recursive functions of control, without becoming simply incorporated or used by the system (i.e., by transforming dissipative energy into information). Entropic data are operative agents of irreducible size that crack and rescript the source program of the system from within. The system of governance defined by the digital world of data can therefore no longer rely upon the smooth programming of tasks, the exact reproduction of rules, and the optimization of conducts, habits, and behaviors. *Randomness* has become the condition of *programming culture*.

This book does not imagine a world in which rationality has been replaced by the arbitrariness of information. Far from it: computational *randomness* corresponds to infinite volumes of data that are meaningful contingencies which refuse to be fully comprehended, compressed, or sensed by totalities (i.e., by the mind, the machine, or the body). This also means that algorithms do not exclusively channel data according to preset mechanisms of binary synthesis (0s and 1s), as they also enumerate the indeterminate zone between finite states. This new function of algorithms thus involves not the reduction of data to binary digits, but the ingression of random quantities into computation: a new level of determination that has come to characterize automated modes of organization and control. Far from making the rational system of governance more efficient, this new level of determination forces governance to rely on indeterminate probabilities, and thus to become confronted with data that produce alien rules. These rules are at once discrete and infinite, united and fractalized.

From another standpoint, the emphasis on the new tendencies of algorithms to be overshadowed by infinite volumes of data explains the ingression of computational logic into culture. What is important here is not that culture has become doomed by the automated rules that transform its variety of expressions into data that can be classified, profiled, and consumed. Instead, the addition of random quantities to finite procedures turns automation into a computational adventure resulting in the determination of new cultural actualities. Instead of being exhausted by the formalism of rules or symbols that execute instructions, automated processing requires a semiopen *architecture* of axioms, whereby existing postulates are there to be superseded by others that can transform infinite quantities into contingent probabilities. Incompleteness in axiomatics thus brings to light the fact that automated processing is not predeterminate, but rather tends toward new determinations. In making this claim I do not intend to suggest that computation can now explain culture, aesthetics, and thought because it can account for change. My contention is rather that there is a concrete culture, an aesthetic and a mode of thought, specific to the computational production of new probabilities.

This is why this book argues for a new digital space that no longer or not fully coincides with Deleuze and Guattari's notions of "striated" (metric) and "smooth" (vectorial and projective or topological) space. Striated space is gridded, linear, metric, and optic.[2] It is also described as the space of logos, based on the deductive reduction of infinities to discrete unities constituting the building blocks of reason, the function of which is to find solutions to occurring problems.

In this book the striated space corresponds to the digital matrix of points that do not change over time: a prefixed, gridlike architecture derived from postulates based on discrete sets of algorithms through which optimal forms can be constructed. This is the striated space of the city, the urban planning deduced from the exact relation between points, which establishes an infrastructural grid that predetermines movement.[3] In the last twenty years, however, the digital mapping of space has been intersected by a new tendency in digital design that has more fully embraced the power of computation to generate new architectural forms or smooth surfaces. By drawing on biological notions of morphogenesis, and thus by relying on the capacity of forms to change over time, algorithms have become generative components for form-finding and pattern-making architectures. The new centrality of generative algorithms (but also cellular automata, L-systems, and parametricism) in digital design has led to the construction of various topological geometries and curvilinear shapes that have come to be known as blob architectures. While the gridlike architecture of striated space (or digital mapping) places discrete unities at the center of a design made of points connected by lines, the topological curves of smooth space (or blob architecture) starts from the generative power of a point, the meshing and folding of which becomes the condition for the emergence of a new form.

Far from being in direct opposition, Deleuze and Guattari often refer to these two spaces as being in a relation of reciprocal presupposition, so that points can generate new curves, and curves can become frozen segments. However, this mutualism between the two kinds of spaces—or planes—may not be fully sufficient to explain the mode of extension produced by the ingression of computation into culture. To the striated (metric) and smooth (topological) spaces, this book annexes another approach to extension. This approach is defined by *mereotopology*: the study of the relation between parts, of that between parts and wholes, and of the boundaries between parts. In particular, I turn to Alfred North Whitehead's schema of mereotopological relations—a schema that is a concrete abstraction—in order to argue that neither discrete unities nor continual surfaces can account for the transformation of the digital grid, as the latter is characterized by the infiltration of randomness into finite sets of rules.

Mereotopology describes parts as being semiopen: it casts them as discrete and separable on the one hand and as undivided and continuous on the other. It postulates that there is no gap between parts, and neither are there *infinitesimal* points constituting continuous trajectories (or topological surfaces). Instead, between points there are always more points (or an infinite amount of points), which correspond not to infinitesimals, percepts, and affects but to finite segments internally defined by a unique arrangement of infinities. For Whitehead these finite segments are actualities, which are at once extended and intensive, or equipped with space and time; they are finite durations. In contrast to blob architectures, which have given rise to a computational aesthetics expressed by the topological surface or the smooth plane of total connection, mereotopological architecture reveals that infinity is intrinsic to parts, unities, and discrete objects. From this standpoint, infinity does not coincide with the total fusion of spatiotemporal dimensions into one deforming surface, but instead can be explained by how wholes (continuities) become parts (discontinuities), and how parts can be bigger than wholes. In computational terms, infinity is equivalent to random (or incompressible) quantities of data (which are at once discrete and continuous) interfering with and reprogramming the algorithmic procedures in digital design, for instance. This also means that algorithms are not the building blocks of a topological surface whose forms continuously evolve. What connect the multiplicity of points are instead infinite quantities that ingress into the gap between points, thereby revealing the existence of yet another point (or spatiotemporal actuality) that overlaps them, but which does not originate from them. Yet how do these quantities come to determine and characterize algorithmic procedures in digital design?

This is where computation becomes entangled with Whitehead's view that it is *prehensions* that define what an entity is and how it relates to others. Prehensions point to how any actuality (from an animal body to a grain of sand, from an amoeba to an electron) grasps, includes and excludes, and transforms data. Instead of an ontological dominance of higher forms of actuality (such as human beings) over others, Whitehead argues that all entities have an equivalent status. Not only are they all real, but also they all matter. Nevertheless, this seemingly flattening ontology does not simply contend that these actualities are all the same, nor does it hold that they are all different. Whitehead proposes a radical pragmatism according to which determinate events, or what he calls occasions of experience, are defined by degrees of prehension that in turn constitute the degree of importance of some actualities compared to others.

In this book, the new function of algorithms within the programming of spatiotemporal forms and relations reveals how the degree of prehension proper to algorithms has come to characterize computational culture. Algorithms are no longer seen as tools to accomplish a task: in *digital architecture*, they are the constructive material or abstract "stuff" that enables the automated design of buildings, infrastructures, and objects. Algorithms are thus actualities, defined by an automated prehension of data in the computational processing of probabilities. From this standpoint, digital algorithms are not simply representations of data, but are occasions of experience insofar as they prehend information in their own way, which neither strictly coincides with the binary or fuzzy logic of computation nor with the agency of external physical inputs. Instead, as actual occasions, algorithms prehend the formal system into which they are scripted, and also the external data inputs that they retrieve. Nevertheless, this activity of prehension does not simply amount to a reproduction of what is prehended. On the contrary, it can be described as a contagion. This is because to prehend data is to undergo an irreversible transformation defined by the way in which rules are immanent to the infinite varieties of quantities that they attempt to synthesize. This means that rules cannot change these infinite quantities; instead the latter can determine rules anew and thus produce new ones. From this standpoint, I do not use this notion of contagion to suggest that there is a physical connection between points (i.e., that one point of prehension is determined by the next point in a sequential order) or a potential relation between points (i.e., the fact that points are linked by infinitesimal approximations). Instead, to maintain that a prehension can be understood as a contagion is to say that infinite amounts of data irreversibly enter and determine the function of algorithmic procedures. It follows that contagion describes the immanence of randomness in programming. This irreversible invasion of incompressible data into the digital design of space has led to the production of digital spatiotemporalities that do not represent physical space, but are instead new spatiotemporal actualities. The contagious architecture of these actualities is constructing a new digital space, within which programmed architectural forms and urban infrastructures expose not only new modes of living but also new modes of thinking.

Nonetheless, by prehending (or becoming infected with) infinite quantities of data, algorithms do not simply work to generate optimal probabilities that will more closely match the architecture of the future and its urban infrastructure. The futurity of algorithmic prehensions cannot be exhausted by the image of the future. Instead, as prehensive entities, algorithms unleash the concrete futurity of the digital spatiotemporalities of the present, of which digital architecture is but one example (other examples might include the relational architecture of databases, the cultural, political and economic statements of search culture, the connectedness of social media, and the immediacy of data communication).

This book is about the ingression of computational logic into culture. It is most appropriately placed in the field of digital architecture, because the algorithmic production of digital spatiotemporalities defines: (1) that logic is becoming an aesthetic operation, and (2) that computational aesthetics is characterized by the algorithmic prehension of *incomputable* data. In adding this aesthetic interference to computational logic I do not mean to imply that algorithms are the new synthesizer of indeterminate quantities. On the contrary, one condition of this book is that no actuality—physical or automated—could ever contain the infinite amount of infinities that are immanent to all actualities. Instead, what happens with all actualities is that these varieties of infinities are only partially and uniquely processed, so that not only is each actuality asymmetric with respect to another, but it is also asymmetric within itself. In other words, the discovery of incomputable quantities in axiomatics reveals that there can never be any totality that could subsume (external or internal) parts into one encompassing whole.

From this standpoint, the aesthetic operations of logic suggest that the prehensive activity of algorithms not only evaluates and transforms, but also enumerates and produces new computational actualities. In the field of digital architecture, this means that computational logic does not need to be used to reach aesthetic results as if it were operated by an external agent, which would select the activities of the process from an "outside." Aesthetics must instead be understood to reside at the core of computational logic, because it defines computational processing as the determining of infinities in a step-by-step fashion, and without subjecting them to complete synthesis and/or axiomatics. Aesthetics, that is, is not only complementary to logic but is immanent to it: it exposes contingency in programming, and the reality of chance in the calculation of probabilities.

It would be misleading, however, to attribute the aesthetic capacities of algorithms to a mainly qualitative synthesis of data. It is important to bear in mind when speaking of aesthetics in computation that one cannot obviate the entropic size of data, and therefore the tendencies of quantities to increase in volume, length, and density each time they are calculated. Thus, this book does not depart from one basic crux of computation: namely, the fact that computation is a method of quantification that deals with quantities. From this standpoint, *algorithmic prehensions are quantifications of infinite quantities that produce new quantities*.

This is also to say that there is a production of the new within computation that specifically concerns increasing randomness or increasing volumes of data that cannot be systematized in smaller algorithmic procedures. This book therefore contains no claims as to the necessity of cleansing culture of data pollution, because it admits that data production is an immanent process that unravels the gaps, blind spots, and incompatibilities within formal systems in their attempt to constantly invent new axioms and rules.

Similarly, this book also distances itself from the dominant cybernetic model of feedback control, which aims to include qualitative data in computational procedures by allowing the system to become co-constituted by its outside. In particular, the dominance of *second-order cybernetics* and its autopoietic model of feedback in digital architecture has led to a plethora of interactive projects whereby algorithms are designed to respond and adapt to external inputs, so as to be able to add chance to programming. Yet rather than challenge computation, this attempt to add qualitative data to programming has in my opinion served to reify the fundamental system of inference which assigns logic to rationality and aesthetics to sensation. Against this tendency, this book embraces the aesthetic function of algorithms in their quantitative concreteness, the prehension of contingency and thus the outbreak of randomness within logic. I claim that this is the computational aesthetic that governs digital culture today.

The investigation of this weird formalism points to a further level of analysis that looks for the properties of a speculative function of computation. In doing so it turns again to Whitehead's metaphysics, because the scope of his attempts to disentangle reason from the enclaves of rationality are sufficiently broad to include the possibility that automated modes of thought are modes of decision, and that decision is a mode of adding new data to and thereby rethinking what already exists, by counteracting the sequential order of patterns. In short, Whitehead's study of the function of reason has offered my investigation the opportunity to discuss a mode of thought proper to algorithms: *soft(ware) thought*. Instead of looking for ways of comparing (or conflating) computation with (or into) formal or practical notions of reason, and instead of thus associating it with conceptions of the mind that view the latter as something that executes thoughts onto the world, or as something produced by the synaptic connections of the brain's neural networks, my analysis starts from the reality of algorithms as actual modes of thought.

Seen from this standpoint, computation does not refer to a rational calculus that deduces reality from universal axioms, but rather to the algorithmic prehension of the random data that are now contaminating formal logic's attempts to continuously invent new axioms. The speculative function of algorithms corresponds to an abstract scheme of concrete data, or to enumerations of procedures through which computation is constructing our present. Thus, speculative computation is not to be confused with a new mode of prediction, which for instance forecloses the potential threat of the unknown by prompting immediate decisions that anticipate the happenings of the present. Instead, this abstract scheme includes interference (the entropic expansion of quantities) in the procedures of the present insofar as it allows infinite volumes of data to determine spatiotemporal activities. In short, speculative computation is not a new system of probabilities that tries to turn potentialities into possibilities. It is instead an aesthetic ordering of entropic data. This is a weird ordering that involves the prehension not simply of temporal infinities but also of the infinities of extension as they become enumerated in computational procedures. This immanent partiality helps us to describe computation in terms of Whitehead's *speculative* function of *reason*, according to which algorithmic actualities select quantitative novelty from repetition, thereby allowing computation to add new data structures or spatiotemporalities to the extensive continuum of actualities.

This speculative character of computation cannot be accommodated by a cybernetic system of probabilities. As Massumi has clarified in his discussion of the efficacy of preemptive power, such a system can no longer rely on already processed data. Instead, the cybernetic mode of control based on feedback as the self-regulating property of governance—whereby the output allows the system to incorporate more complexity, and thus to become extended into or fused with its outside—now needs to account for what is not there, i.e., for the determinacy of the unknown. This is why the binary language of digital computation is no longer sufficient to anticipate the emergence of errors, or to convert unknown quanta into preset probabilities. Thus, as Massumi explains, the cybernetic apparatus of control, which is based upon and defined by the operation and the operability of procedures, employs a quasi-empirical mode of calculation, according to which the necessary emergence of the new (the uncertain) and its potential effects are precalculated and preempted before the fact. In other words, the effects of the unknown have become the causal motor by which control is unconditionally exercised and driven by immanent decisions about what has not yet happened.

The cybernetic system of feedback therefore inserts temporality or qualitative variations into its binary calculation. In particular, the calculation of infinitesimal variations between these states has challenged cybernetics to overcome its own limit, and thereby to extend its power of prediction toward qualitative variations. It is this stretch toward the inclusion of temporal variations that reveals postcybernetic control's power to act retroactively, i.e., to act by turning the potential effects of the future into operative procedures within the present. The matrix of binary digits is therefore turned into a fold of approximate calculations of the infinitesimal points that join two coordinates at a tangent: the derivatives of the *x* and *y* coordinates turn parallel lines into the infinities of a potential curve. A *topological* surface thus rises above the digital matrix of sequential coding, and is ceaselessly reproduced in the digital design of facades, buildings, and urban planning. This computational aesthetic of the curve is now the dominant expression of postcybernetic control.

My investigation however does not stop at this point, as it continues to explore the stubborn reality of quantities that remains at the core of digital architecture. The reason why algorithmic and interactive architecture—or digitality in general—has been unable to grasp or produce the intensive qualities of spatiotemporal experience, of the bodily feeling of spatiotemporal variations, is that computation deals in quantities and quantifications. This book asks the reader to consider the density of computational quantities as enumerations of new actualities, or spatiotemporal entities that enter and are added to the infrastructural organization of information. The book thus embarks on a close exploration of digital architecture projects in order to account not for the generative evolution of a topological surface, but rather for the *mereotopology* of parts that are bigger than wholes. Here, once again, algorithms are foregrounded as actual occasions of data that cannot be subsumed under the totalizing framework of postcybernetic control. These parts, I suggest, do not become the fused agents in a smooth space of control: they are instead autonomous events or nexuses of actual occasions.

The mereotopological exploration of computational quantities leaves my investigation with yet another question to discuss. If digital architecture implies the production of computational space-time, does it follow that there is an architecture of thought proper to computation? The pursuit of this question leads the book's arguments toward the inevitable realization of the incomprehensible existence of soft thought: an automated mode of prehension that cannot be compressed into a totalizing system (i.e., the mind, the machine, the body, or into idealism, mechanicism, or vitalism).

Soft thought is not the new horizon for *cognition*, or for the ontological construction of a new form of rationality. Instead, soft thought stems from the immanent ingression of *incomputable* data into digital programming. Soft thought is not what replaces thinking understood as a cognitive action, or affords the mind new capacities to order and calculate, or indeed gives the body new abilities to navigate space. Simply put, soft thought pertains to the existence of modes of thought, decision making, and mentality that do not exist in direct relation to human thinking. These modes of thought (of which soft thought is only one configuration) maintain a certain degree of autonomy from cognition demonstrated by their logical inconsistencies. This book thus ends with no surprise or final revelation, but with one remark: soft thought is not there to be understood as a new cognitive function or as a transcendent form of rationality, but to reveal that *programming culture* is infected by incomputable thoughts that are yet to be accounted for.

1 Incomputable Objects in the Age of the Algorithm

The late twentieth century may one day be known as the dawn age of the algorithm. If so, we wish to be the first to embrace the new rationality that sees space and matter as indistinguishable, as active mediums shaped by both embedded and remote events and the patterns they form.[1]

1.0 Metamodeling

Algorithms do not simply govern the procedural logic of computers: more generally, they have become the objects of a new programming culture.[2] The imperative of information processing has turned culture into a lab of generative forms that are driven by open-ended rules. Whether we are speaking about DNA, bacteria, or stem cell cultures, cultures of sounds and images, time-based cultures, or cultures of spatial modeling, algorithms now explain evolution, growth, adaptation, and structural change. Algorithms, therefore, have become equated with the generative capacities of matter to evolve.[3] It is not by chance that the age of the algorithm has also come to be recognized as an age characterized by forms of emergent behavior that are determined by continual variation and uncertainty.

Computational studies of evolutionary processes have taken to modeling randomly mutating software instead of creating large, complex simulations of biological systems. For example, the new field of metabiology aims to develop all possible designs for biological organisms within a software space.[4] These metabiological designs are examples of an *algorithmic architecture* that appears to deploy the software of matter itself.

But if generative algorithms[5] are no longer mere simulators of material dynamics, it may then be possible that they have acquired a new, ontological status that is unrelated to the preexistence of biophysical bodies. In other words, the mode of existence of algorithms no longer merely corresponds to models that simulate material bodies: instead it constructs a new kind of model, which derives its rules from contingencies and open-ended solutions. Generative algorithms are said to dissolve the opposition between mathematics and biology, between abstract models and concrete bodies.[6] Just as matter has an abstract form, so too have software programs become evolutionary bodies.

Nevertheless, the ontological claim for an ultimate merging between information and matter does not seem to account for a simple but entirely inevitable question: if, as Kwinter points out,[7] the age of the algorithm announces the advent of a new rationality, in which matter and space become indistinguishable, how can what is abstract remain as real as what is concrete? If the model and matter are fused into *one* invariant principle of continual variation, then what makes novelty take place? What adds novelty to this system of perennial change? These questions will guide this chapter's attempts at defining algorithmic objects as data actualities that exist without and beyond their biophysical referent and mathematical form.

From this standpoint, it is important to discuss the place of algorithms within recent critical approaches to computation and cybernetics.[8] In particular, the view that information systems are open and not closed, dynamic and evolutionary (with rules that change over time), and are thus not preprogrammed, has led most cultural analysis to argue that information, far from being abstract, is always already another form of matter. But although this new version of materialism has contributed toward challenging the assumption that the information model *represents* (i.e., governs) materiality, it has also divorced algorithms from their own reality and undermined insight into the true nature of the latter.

I will argue in this chapter that this form of materialism has led to a naive reading that flattens algorithms onto the biophysical ground, which they are then said to shape. By overlooking the existence of actual entities that cannot be physically felt and cognized, the cultural approach to cybernetics and computation has thus dissolved the reality of algorithms into thin air.

I suggest that a refusal *tout court* of the actuality of algorithms has coincided with the disqualification of computational formalism and the attendant rise of a computation that is driven by biophysical and sensorimotor responses. It will be claimed here that by leaving behind computational formalism (and symbolic logic) and the entropic nature of closed systems, so as to highlight the open-ended and negentropic forces of self-organization, cultural analysis has denied algorithms the potential of being anything other than a finite set of rules.

For this reason, this chapter questions the centrality of the *metacomputational* approach to information theory, according to which finite sets of algorithms, or mathematical *axioms*,[9] generate infinitely complex structures. At the same time, however, the chapter also contests the replacement of systems based on finite sets of algorithms with interactive processes relying on external inputs and temporal variations. The chapter concludes that the ideas about algorithms that are propounded by the metacomputational and the interactive approaches are characterized by their dismissal of the view that algorithms are actual entities imbued with infinity.

In order to challenge such assumptions, it seems important here to rearticulate the notion of the model, so as to distance it from the assumptions that algorithms either constitute the model of biophysical reality or are the latter's result. For this reason, I will turn to a reformulated version of Félix Guattari's concept of *metamodeling*.[10]

This concept can explain that the potential of computation coincides neither with the generative power of algorithms to design self-evolving structures, nor with interactive systems of physical connections. In other words, I suggest that neither the mathematical nor the physical underpinnings of computation can suffice to describe what an algorithmic object can be.

Guattari's concept of metamodeling is important because it challenges both mathematical truths and physical laws. It offers us the opportunity to describe an *extraspace* of nonunifying actualities, a *contagious architecture* that does not prioritize formalism or empiricism. This extraspace of algorithmic actualities is not to be found outside or between mathematics and physics, but rather within mathematical truths and physical laws. This is an extraspace of algorithmic actualities that are infected with abstractions but which are not themselves abstract. To be infected with abstractions means that these are immanent to actualities to the extent that they are intrinsic to their composition and finitude. There is therefore no contagious architecture between algorithms or between algorithms and biophysical bodies. Instead, contagion is taken here to define the quasi-finitude of algorithmic objects: the fact that these objects are spatiotemporal actualities which (and this is specifically discussed in this chapter) cannot be summed up in smaller programs, and which do not result from the sum of their parts. Similarly, it is argued here that this extraspace is not defined by the generative capacities of algorithms, from which rules evolve over time to deploy complex behavior. On the contrary, this extraspace corresponds to the random, incompressible data of algorithmic objects that are immediately experienced as irreducible parts larger than any totalizing whole.

Guattari's concept of metamodeling offers a critique of the notion of the model, but at the same time does not collapse the actuality of algorithms into that of biophysical objects. Instead, Guattari understood models in terms of cybernetic systems. He believed that models were simulations based on patterns of recognition (social, cultural, political, aesthetic patterns of governability).[11] Models, therefore, were for him reductions of a diagrammatic space made of intersections and disjunctions, operated by abstract signs and symbols. This diagrammatic space is a metamodel and not a series of prototypes or inherited and learned behavioral patterns. Instead, for Guattari signs and symbols become layered together without having any direct correspondence. Metamodeling—as opposed to the cybernetics of probabilities based on the possibility of forecasting the future through the data of the past—explains how signs and symbols are probes of futurity engaged in building the invisible architecture of the present. Metamodeling, therefore, describes how process becomes configuration, or how potentialities exceed preordained typologies.

Since the model is a formal structure but also a psychologically inherited blueprint (instructions or codes), Guattari argued that the hierarchy between models and facts, the formal and the practical, had to be turned into an ethico-aesthetics of signs, symbols, and objects. This meant that rules did not have to conform to programs, but needed to be resingularized and reconfigured while being subtracted from their realm of probabilities. In other words, metamodeling described how any set of rules "constructs its own cartographies, its own reference points, and thus its own analytic approach, its own analytic methodology."[12]

For Guattari, mathematics, but also software, can be taken as examples of metamodeling communication, which bypasses the imperative of representation. He argued against the idea that mathematics is the language of physics. Instead, he believed that mathematics articulated material processes of production that could not be physically detected or proven. Guattari's notion of metamodeling suggested that mathematical signs had no physical objects as their referent, but instead described a reality greater than could be physically explained.[13] In particular, he thought that the diagrammatic operations of mathematics pointed out "a physico-mathematic complex which links the deterritorialization of a system of signs to the deterritorialization of a constellation of physical objects."[14]

The intersection of these deterritorialized signs (mathematical symbols deterritorialized from the system of formal language) and objects (deterritorialized from their physical laws) defined metamodeling as the discovery and construction of new worlds and novel actualities. Hence, metamodeling was not simply the result of summing distinct models into a transcendent system/order, or merely the transposition of a model from one field into another (for instance, moving the scientific model of cybernetics into the cultural system of grouping). From the standpoint of metamodeling, the sign and the object both exceeded their mathematical and physical realms by setting up new conditions for change beyond formal schemes and empirical evidence.

Guattari's notion of metamodeling may thus help us to argue that algorithmic actualities can be thought independently of formal models. At the same time, however, it may be necessary to push his notion of metamodeling further, so as to avoid equating it with the logic of mixing and remixing signs and objects through digital computation.[15] This means that it is important to disagree with Guattari's notion that metamodeling announces the arrival of a postmediatic era in which all media forms share the same mathematical language. Instead, metamodeling is used here to suggest that a mathematical model, no matter how deterritorialized it is, cannot fully explain the actuality of algorithmic objects. This algorithmic actuality is transverse to both the mathematical and the physical domains. It is a transverse actuality. Thus, and *contra* Guattari, it will be argued here that algorithms are transverse objects constituted by mathematical and physical limits, the points at which patternless or incomputable data have infected all mediatic forms of communication.

These patternless data define not a new kind of algorithmic matrix so much as the immanence of incompressible data in all diagrammatics. As will become clearer below, algorithmic objects are both actual and abstract entities. In a manner that differs from Guattari's diagrammatics and metamodeling of nonsignifying connections, it is suggested here that in order to define algorithmic objects one has to admit the possibility that there may be an extra layer of potentiality within axiomatic computation,[16] a layer that is not exclusive to the empirical realm. It is this extra layer of potentiality—the reality of abstract objects—that this chapter intends to argue for.

But how, when discussing algorithmic objects, can one articulate this potentiality? And hasn't this potentiality already been discussed in computation, as that which temporarily appears in the foreground of experience as an evanescent object, but which then withdraws from actual existence? Isn't an emphasis on the abstraction or on the potentiality of algorithmic objects just another way to say that they only appear temporarily within a range of given possibilities, without ever existing as actual entities (i.e., a sterile rehearsal of Platonism)?

While I agree that algorithmic objects are not physical entities, it is difficult to deny that sets of instructions are actual data. One obvious example, and one that will become important here, can be found in the fact that algorithms correspond to data objects that build actual instances of space and time that have volume, weight, gravity, depth, height, and density, and can thus be found in algorithmic architecture. I suggest that algorithmic architecture explicitly offers us the opportunity to understand that these data objects are not simulations of some biophysical ground of the past or future. Algorithmic architecture corresponds to the software production of space as it implies automated data able to evolve in a search space so as to design buildings, urban infrastructures, and city plans. In algorithmic architecture, algorithms are not exclusively defined by the quality they can reproduce (color, sound, or variables), but also by the quantities of data that they operate. In other words, algorithmic architecture cannot overlook the fact that algorithms are quantifications of data that are at once actual and abstract and thus cannot be reduced to one plane of reality (that is, they cannot be flattened down into one continuous plane of qualitative relation). These quantities are not characterized by external relations (*partes extra partes*; i.e., one part of space is exterior to another part) but need to be understood as infinite varieties of parts that infect (take over and program) actual algorithmic sequences. Actual algorithms are indeed the hosts of abstract quantities or infinite varieties of infinities that constitute, at the same time, both condition and limit of any actual entity as a finite set of instructions. As actualities, algorithms hold within themselves abstractions that determine both their infinite change and their present finite status. As abstractions, algorithms are defined by the internal relations of an infinite variety of infinities. Abstract algorithms are unrelated to one another and can only enter into contact with one another once they are hosted by actual algorithms or finite computational states. I therefore claim that algorithms are both actual and abstract. They are actual and thus spatiotemporally determined, conditioned and limited. But they are also abstract, and are thus capable of irreversibly determining change according to the degree to which they contaminate actualities.

While algorithmic architecture explicitly treats algorithms as data objects, it has also failed to articulate their existence and has equated them to empty abstractions and/or simulations of physical entities. Similarly, the debate between metacomputational structures and interactive and responsive space still lacks a thorough engagement with algorithmic objects. On the one hand, algorithmic architecture has rearticulated formal mathematical structures in terms of evolving dynamics such as *cellular automata*. On the other hand, the shift toward the use of interactive algorithms has led designers to rely on biophysical inputs. Algorithmic architecture, therefore, has not offered a solution to the question of temporal objects, or to the problem of a biophysically induced computational structure. It views algorithmic objects either as the result of evolving data patterns or as a reaction to biophysical inputs. But my aim in this chapter is not to look for a solution to a problem. On the contrary, I am suggesting here that the questions of *what* an algorithmic object is and indeed of *how* it is have not been sufficiently discussed. This chapter intends to remedy that situation.

Nevertheless, in order to embark upon this project, it is first of all important to engage with computational theory, according to which a question of infinity lies at the core of the algorithm. Since the invention of the Turing machine, the problem of computing infinite quantities of data (or abstract quantities) into finite sets of rules has blended with the more general problem of ordering and programming noncomputable (incomputable) algorithms. Viewed from this standpoint, the debates within information theory that pertain to this problem already reveal to us that the ontology of algorithmic objects is to be found within the incompleteness of the axiomatic method.[17] Similarly, if one does not engage with the challenges posed to the axiomatic underpinning of algorithmic objects, then it becomes impossible to address their uniqueness and singularity.

However, it is important to be cautious here. This chapter does not argue that a thorough mathematical investigation of algorithms will give us answers to the questions noted above. On the contrary, I use algorithmic information theorist Gregory Chaitin's notion of *Omega* (incomputable algorithms at the limit of any computational process) to argue that any search for a mathematical Holy Grail is completely futile. Chaitin's theory about incomputable probabilities suggests that any closed set of finite algorithms is imbued with incompressible data, and by taking this notion further, beyond the specific field of algorithmic information theory, I argue that algorithmic objects cannot be contained by a metacomputational ontology.

Similarly, I also question the assumption that only what lies outside the realm of computation, automation, discreteness, and finitude can help us to define algorithmic objects as dynamic, changeable, and in movement. Instead, the concept of Omega proves that although infinity cannot be found in the physical world, it can be discerned within computational processing. This means that the possibility for change is intrinsic to the ontology of algorithmic actualities. Consequently I argue that interactive algorithms and responsive computation (i.e., algorithms defined by external factors, or driven by external inputs) do not contribute to the explanation, but rather to the occlusion, of the *what* and *how* of algorithmic objects. In particular, this chapter points out that interactive and responsive architecture end up attributing change to external agents, actuators or participants; algorithmic objects are thus seen as remaining passive in the face of an ever-changing environment of interaction.

The chapter demonstrates that computation offers us a rather more complicated and subtle notion of algorithms, according to which the latter are not equivalent to evolving agents that mutate in time, but are sequential spatiotemporal data structures conditioned by an infinite amount of information. These data structures are actual spatiotemporalities and have precisely become the objects of algorithmic architecture. In chapter 2 these structures will be analyzed more closely; here my aim is to develop an understanding of algorithms *qua* actual entities.

To engage further with the ontology of algorithms, one cannot avoid discussing the philosophical problems of what constitutes an object, and whether or not there are such things as abstract objects. In order to address these problems I will draw on Alfred North Whitehead's process philosophy, and on Graham Harman's object-oriented metaphysics. The encounter between these two contrasting metaphysical systems will also be used to challenge the idea that algorithmic objects are finite sets of instructions. This discussion is intended to shed some light on why the notion of the incomputable is able to help us to define algorithms in terms of actual and abstract objects.

This chapter will conclude that algorithmic objects are actual entities: spatiotemporal structures imbued with incomputable or patternless objects. The latter are not, however, to be misunderstood as the indefinite background of self-evolving energy. On the contrary, patternless objects correspond to entropic bursts of energy within sets of instructions, thereby defining the odd existence of discrete yet infinite algorithms within the structure of our programming culture. In this sense, algorithmic objects are not simply emergent forms within software, but are discrete unities injected with random data. It is as if algorithmic objects no longer pertain to the realm of software, but have unintentionally built an extraspace of data that infects (or irreversibly reprograms) all levels of matter.

From this standpoint, an algorithmic object is more than a temporal appearance or the result of interactive stimuli. Instead, it is a symptom of the new spatiotemporal structures that are most clearly deployed by algorithmic architecture. This is an important point against the idea that algorithms are merely temporal forms that are destined to disappear in the background of ubiquitous computation. Computational design problematically embraces the logic of prediction and the calculation of probabilities, and is unable to explain novelty in spatiotemporal experience. I argue against this form of metacomputation, and against the rational logic based on few unchangeable rules, the combination of which is held to produce all forms of complexity.

Contrary to the view of computation as a form of rationalism, I will suggest here that the ingression of the incomputable in axiomatics leads us to rethink computation in terms of speculative reason, to borrow from Whitehead. Computation, it will be argued, is an instance of speculative reason, since it no longer nor exclusively aims at the prediction or calculation of probabilities. On the contrary, Whitehead's understanding of speculative reason explains that the function of reason is to add new data to what will always already happen in an efficient chain of cause and effect. Similarly, it will be observed that the speculative view of computation implies that calculation is not equivalent to the linear succession of data sets. On the contrary, and as will be explained later in the chapter, each set of instructions is conditioned by what cannot be calculated: by the incomputable algorithms that disclose the holes, gaps, irregularities, and anomalies within the formal order of sequences. This means that a notion of speculative computing is not concerned with quantifying probabilities to predict the future, but with including random or patternless quantities of data in sequential calculation so as to add novelty in the actual architecture of things. This is why a notion of speculative computing is not to be confused with the capacity of algorithmic architecture to create temporary forms. On the contrary, the notion of speculative computing advanced here suggests that random data—indeterminate quantities—are the contagious architectures of the present. These architectures, far from withdrawing from actuality and thus being temporal forms that appear and disappear, rather remain actualities: spatiotemporal realities which are objective data to be inherited, evaluated, and appropriated by future actualities, of whichever kind at whichever scale, even when they cease to be there.

Far from predicting the spatiotemporal structures of a possible future, algorithmic architecture is conceived here as a symptom of the speculative programming of the present. Algorithmic architecture can also count as an instance of computational aesthetics, understood here in terms of the algorithmic prehension[18] of indeterminate data. Computational aesthetics therefore is not about the idealism of form, or about a code unraveling the complexity of biophysical structures. Instead, this chapter will conclude that algorithmic architecture explains computational aesthetics as the programming of actualities through the algorithmic selection of patternless data. For this reason, algorithmic architecture is another form of *postcybernetic control*, because it relies on algorithms to prehend incomputable data in order to program culture.

1.0.1 Programming the living

It is hard to understand what is meant by the age of the algorithm without referring to at least two distinct conceptions of algorithms. On the one hand, algorithms correspond to a set of finite instructions. On the other, algorithms have been conceived as evolving data able to adapt and to vary unpredictably according to external stimuli.[19]

A cybernetic reading of computation may clarify these two points. From the standpoint of *first-order cybernetics*, computation is a closed system, a formal language able to describe any biophysical process without having to be acted upon by the external environment. This is a closed, self-sufficient set of programmed instructions able to predict the future behavior of the system in terms of preset probabilities. On the other hand, *second-order cybernetics* suggests that biophysical indeterminacy or the contingency of environmental factors can open software programming to the modeling of dynamic systems that change over time and generate results that differ from initial conditions.

It is not argued here that these two tendencies simply correspond to a historical shift, whereby algorithms are no longer to be understood as finite sets of data but as interactive instructions open to change. Similarly, it may be misleading to assume that the age of the algorithm and the advance of programming cultures can be described simply in terms of an epistemological shift (and progress) from symbolic formalism to a biophysical understanding of computation and information systems. To embrace this view of an epistemological shift is to imply that algorithms lack the irregular dynamics of biological or physical systems. The second-order cybernetics understanding of computation certainly points out that dynamics or change can only be derived from the indeterminacy of living systems. Yet I will argue here against this predominant tendency, as it discards the possibility that change could concern the formal logic of computation, and by doing so forces the abstract reality of algorithms to become a mere effect of the biophysical world.

In the field of algorithmic architecture, one recent example of such a tendency can be found in the works of architect Greg Lynn, whose digital design takes inspiration from biophysical vector fields that lead to the growth of an emerging algorithmic form. This approach to design sees architectural form as the result of the computational processing of biophysical variables (e.g., the distribution of weight, gravitational pressures, the circulation of air, the movement of people). By closing the gap between mathematical models and biophysical contingencies, second-order cybernetics has turned computation into a temporal system, which explains change through the iteration of codes into a search space for evolving complexity. From this standpoint, biophysical unpredictability has become superior to mathematical calculability, and the reality of abstraction has slipped behind the concreteness of matter.

According to architect Karl Chu, algorithms have been central to the late-twentieth-century convergence of computational and biogenetic revolutions leading to the ultimate design of biological and mathematical codes, which promises the embodiment of life, emotion, and intelligence through "abstract machines or through biomachinic mutation of organic and inorganic substances."[20] Of course, one cannot deny that this biodigital combination of material parts arranged by algorithmic computation has added a distinctive trait to our information-based technoculture. For instance, much debate about cybernetic machines and biotechnologies in the late 1990s directly engaged with the no-longer-natural essence of biology, and with the new technoscientific ontologies of biological bodies. Since the natural ground upon which biology could be distinguished from artificial technics (for instance evolutionary technics of breeding, reproduction, cross-pollination, adaptation, etc.) was dissolving, it was argued that nature itself was the result of the plasticity of biological forms.

Second-order cybernetics has explained computation in terms of an evolving system that depends on its structural coupling with the environment. According to this biological view of computation, an interactive culture of response drives the calculation of mathematical probabilities. As a result, algorithmic architecture started to adopt these dynamics of biophysical variations in order to provoke changes in the program, for instance by inserting errors into the sequence of algorithms. Similarly, and perhaps more problematically, algorithmic architecture has also been related to a body that acts “as the framer of spatial information, as the source of its ‘autonomy’ or ‘interiority’.”[21]

Against the digital design of space, Mark Hansen argued that architecture “must reconceive its function for the digital age as the art of framing par excellence, it must embrace its potential to bring space and body together in . . . a ‘wearable space’.”[22] Here the algorithmic programming of space is conceived as imposing a coded invariant on the liveliness of an embodied experience of space. A haptic (bodily centered and not merely optical) interaction with software is seen here as necessary for the opening of digital architecture (algorithmic probabilities) to unpredictable variabilities and to the movement of space. This plunges computation into the actualities of biophysical change and novelty.

According to Hansen, algorithmic architectures are unable to explain the biophysicality of space: mathematics can only reduce the biological and physical complexity of living bodies to elegant formulas, and cannot explain the changing nature of experience. Hansen’s objections to the computation of space can be seen as only one symptom of a more generalized critique of computational culture, which opposes codes, rules, and software programming to physicality, change, and indetermination. From this standpoint, biodigital computation must mean the biunivocal relation between mathematics and biology, the fusion of model and execution, the hybridization of the abstract and the concrete, algorithmic models and biophysical space. It has become evident, however, that the structural coupling between programs and bodies has meant an all-encompassing rejection of the reality of abstraction and of algorithmic objects.

It is not by chance that in the last ten years digital media art and architectural projects have worked to annex algorithmic programs to physical sensors. These sensors have the role of gathering data from the environment through actuators, which are set to feed information back into the program. The result is a series of automatic steps that translate control signals into action through motor, light, and speaker capacities.[23] By retrieving sensory data and processing their dynamic value through a string of zeroes and ones, the machine is able to physically interact with the participant. In principle, these procedures will eventually enable the machine (or the software program) to learn and calculate the probabilities of a similar scenario before it actually happens. The cybernetic logic of forecasting the future through models of the past is geared here toward a new level of predictability. It now involves the capacity of the software to rewrite the rules that it was programmed for, and that of the mathematical model to change as a result of physical interactions.

From this standpoint, interactive design has quickly replaced the assumed coldness of finite sets of rules with the ever-present warmth of sensory data. In other words, second-order cybernetics has added biophysical contingencies to the formal language of mathematics. Just as the programming cultures associated with AI defined algorithms through the first-order cybernetics and formal logic of zeroes and ones, so too has the culture of interactive architectures locked itself within second-order cybernetics: the autopoietic self-organization of biophysical systems. What is missing from this picture is a serious consideration of the residual power of algorithms, the processing of rules and the indeterminacies of programming, which are able to unleash novelty in biological, physical, and mathematical forms.

It is impossible to deny that algorithmic architecture has, over the years, become a computational system that has learned to incorporate, run, and anticipate the evolutionary capacities of all material phenomena. Similarly, despite its attempts at merging information and matter, algorithmic architecture has ultimately shown that the reality of abstraction is irreducible to the properties of living systems. To put it in another way, algorithmic architecture has been unable to do away with a problem proper to computation: the problem of calculating infinite series of probabilities in a manner that also includes—and this is significant here—the probability of incomputability.

This power of calculating indeterminacy corresponds not to the merging of information and matter, but to the challenge that the limit of computation has posed to the axiomatic method in computation. According to this method, finite sets of algorithms are programmed to calculate an infinite amount of information, and thus should also be able to compute the future. Nevertheless, this kind of computation could not rely on already-set probabilities that are always already destined to reach a limit beyond which computation would fail, and the prediction of the future would remain a repetition of the past. On the contrary, the computation of the future could only rely on the computational limit. Alan Turing already encountered the problem of the incomputable and attempted to transform the limit of computation into an algorithmic probability. However, as will be discussed in the next section, Turing could not prove the completeness of axiomatics through computation; on the contrary, the problem of random quantities of data continues to haunt computation today. But this is not a problem that solely concerns the field of computation: it is neither a discipline-related problem nor merely a technical difficulty. Instead, I will argue that this problem is intrinsic to all forms of digital programming. More importantly, it is an ontological problem, and one that characterizes computational culture today. This is also to say that in order to define an algorithmic object, one cannot overlook the limit that is imposed upon computation by random data, and one must recognize this as a problem intrinsic to the logic of calculation. To that end, a direct engagement with the problem of the incomputable within computation, framed in the context of algorithmic architecture, will clarify how abstract algorithms are constructing the actuality of spatiotemporal experience.

From this standpoint, the search for the indeterminate, the unpredictable, and change does not need to be grounded in biophysical matter. On the contrary, I argue that the probability of randomness (or incomputable data) is the condition of computation. This is not to assert that there is an underlying mathematical truth able to explain change solely through information. Rather, I suggest that a pure axiomatic method in computation needs to be radically challenged and contrasted with the axiomatic reality of random data.

At the same time, however, a too-rapid retreat into the enclaves of biophysical matter may result in the mere substitution of one problematic solution for another, the risk being that of obliterating the actuality of information and its own indetermination altogether. Instead, in this chapter I point out that the randomness of data is at the core of computation, and yet that these data cannot be fully explained in either mathematical or physical terms. As will be discussed in the next section, these data are "quasi-mathematical," as they can be formalized as probabilities and yet remain incomplete. It is to this apparently paradoxical condition of "incomputable probability" that we will now turn.

1.0.2 Random probabilities

In order to appreciate the role of incomputable algorithms in computation, it is necessary to refer here to the logician Kurt Gödel, who challenged the completeness of the axiomatic method by proving the existence of undecidable propositions within logic.

In 1931, Gödel took issue with mathematician David Hilbert's metamathematical program. He demonstrated that there could not be a complete axiomatic method according to which the reality of things could be proved to be true or false.[24] Gödel's "incompleteness theorems" explained that propositions might be true but could not be verified by a complete axiomatic method. Propositions were therefore ultimately deemed to be undecidable: they could not be proved by the axiomatic method upon which they were hypothesized (instead certain propositions needed new sets of axioms to be added to the original ones).

In Gödel's view, Hilbert's quest for an ultimate algorithm able to reach a finite statement, true or false, pertaining to the initial predicative formula on which it originated proved vain. The problem of axiomatic incompleteness instead affirmed that no decision, and thus no finite rule, could be used to determine the state of things before things could run their course.

Not too long after, the mathematician Alan Turing encountered Gödel's incompleteness problem while attempting to formalize the concepts of algorithm and computation through his famous thought experiment now known as the Turing machine. In particular, the Turing machine demonstrated that problems that can be decided according to the axiomatic method were computable problems.[25] Conversely, those propositions that could not be decided through the axiomatic method would remain incomputable. To put it otherwise: beyond the axiomatic method or the mathematical program that could calculate all (in which all can be decided on the basis of its mathematical ground), Turing realized that there was a constellation of undecidable, incomputable propositions, the reality of which could not be empirically proven. According to Turing, there could not be a complete computational method in which the manipulation of symbols and the rules governing their use would realize Leibniz's dream of a *mathesis universalis*.[26]

For Turing, the incomputable determined the limit of computation: no finite set of rules could predict in advance whether or not the computation of data would halt at a given moment or whether it would reach a zero or one state, as established by initial conditions. This halting problem meant that no finite axiom could constitute the model by which future events could be predicted. Hence, the limit of computation was determined by the existence of those infinite real numbers that could not be counted through the axiomatic method posited at the beginning of the computation. In other words, these numbers were composed of too many elements that could not be ordered into natural numbers (e.g., 1, 2, 3). From this standpoint, insofar as any axiomatic method was incomplete, so too were the rules of computation. As Turing pointed out, it was mathematically impossible to calculate in advance any particular finite state of computation or its completion.[27]

Software artist Alex McLean recently devised a program that addresses the problem of the limits of computation. He called this software the *Fork Bomb Program*, as it served to demonstrate the incumbent software disaster that the Turing machine's halting problem already anticipated.[28] The *Fork Bomb* is a short section of code able to process a large number of algorithmic calculations, which very quickly saturate the available space in the list of processes kept by the computer's operating system. The *Fork Bomb* shows how computational processes are slowed down by means of computational processes. The limit of computation is demonstrated here by the continual repetition of a simple sequence of algorithms, which saturate the search space and make the entire computation collapse.

But if computation has always been haunted by its incomputable limits and by the incompleteness of its axiomatic method, one may wonder how computation has become so central to a culture, which is now characterized as the "age of the algorithm." To understand the cultural (and epochal) dominance of computation, it may be necessary to look at how second-order cybernetics, and in particular the autopoietic understanding of systems, turned the notion of the limit away from ideas of collapse.

Second-order cybernetics and early research in neural networks had already attempted to use the limit of computation to engender self-generating systems of control, the end results of which did not have to match the rules established at their initial conditions. For second-order cybernetics, the autopoietic capacities of biophysical phenomena to self-organize, adapt, and change over time indicated a way out from the cul de sac of research on AI that was strictly based on formal computational models.[29] This meant that the limit of computation could become an evolutionary threshold, to the extent that generative rules could give rise to a new level of order.

If the limit of computation was equivalent to the limit of a physical system whose internal order was threatened by increasing heat or entropy, then the autopoietic model of self-organization demonstrated that chaos could be turned into order in the form of negentropic information.[30] In other words, information could order the energetic chaos of random forces in the same way as biophysical systems self-organize and transform dissipative energy into information. The computational limit thereby no longer posed an obstruction to calculation, but rather became an opportunity for exploring new levels of order able to transform energy into strings of coded instructions.

Paradoxically, however, this new view of computation stemmed from the observation that biophysical systems were able to incorporate uncertain results and account for dynamic structures more effectively than axiomatic methods. This is the view that has constituted the ontological monotheism of evolutionary progress, which searches for increasingly higher forms of order able to minimize (optimize) the entropic randomness of chaos. To put it simply, the shift from formal to evolutionary models of computation has not radically challenged the ontological conception of modeling in itself. On the contrary, the turn toward second-order cybernetics has dismissed what is possibly the most striking implication of computational processing: the arrival of incomputable algorithms in axiomatics.

The early twentieth century's realization that computation was an incomplete affair radically challenged the axiomatic method and forced formal computation to admit that infinite quantities of information had entered its core.[31] If a program is left to run according to precise algorithmic instructions based on the evolutionary drive of growth, change, adaptation, and fitness, then the computational limit arrives as the space of incomputable probabilities that reveal how abstract quantities can reprogram preset rules. The programming of generative algorithms, for instance, does not simply lead to new orders of complexity (in which one level of complexity builds on the previous one, e.g., by transforming entropic energy into useful information), but instead encounters a wall of data that cannot be synthesized in smaller quantities. This wall of incompressible data instead overruns the program, and thus neutralizes or reveals the incompleteness of the axioms on which the program was based in the first place.

These incomputable probabilities are discrete states of nondenumerable infinities. Algorithmic information theorist Gregory Chaitin calls these infinities Omega. The latter corresponds to the halting probability of a universal free-prefix self-delimiting Turing machine.[32] Omega is thus a constant that is computably enumerable, since it defines the limit of a computable, increasing, converging sequence of rational numbers. Nevertheless, it is also algorithmically random: its binary expansion is an algorithmic random sequence, which is incomputable.[33] Hence algorithmic architectures are used not simply to build profiles based on prefixed sets of algorithms, but to exploit the self-delimiting power of computation, defined by its capacity to decide when a program should stop, by transforming nondenumerable infinities into random discrete unities or Omega probabilities: random actualities. These actualities are not simply the product of computation or its representation, but are instead its operative agents, imbued with infinite amounts of data that cannot be synthesized into a smaller, synthetic and complete computational procedure.

However, according to Chaitin, these discrete states are themselves composed of infinite real numbers that cannot be counted through finite axioms. This means that the incompleteness of computational models cannot simply be explained away by the paradigmatic substitution of biological dynamics for mathematical axiomatics. On the contrary, one must explain the incompleteness of computation by addressing contingency within algorithmic processing. This is to say that it is in the axiomatic method of computation that incomputable algorithms reveal the incompressible (infinite, nondenumerable, uncountable) discrete unities, which are strictly neither mathematical nor biological. Incomputable algorithms instead can only be defined by the immanence of infinities in finite sets of data.

This is the sense in which postcybernetic (and postdigital) programming cultures have to be understood: to the same extent that generative algorithms are entering all logics of modeling—so much so that they now seem to be almost ubiquitous (from the modeling of urban infrastructures to the modeling of media networks, from the modeling of epidemics to the modeling of populations flows, work flows, and weather systems)—so too are their intrinsic incomputable quantities building immanent modes of thought and experience.

The age of the algorithm therefore involves the construction of digital space conditioned by incomputable quantities of data.[34] Similarly, it can be argued that programming cultures too cannot be simply modeled through finite sets of probabilities, through which physical variations and approximate profiles can be built. Because real infinities are incomputable, mathematically existing as noncountable by a universal finite axiomatic method, programming cultures are now the immanent hosts of an infinite amount of infinite data. Rather than finite probabilities, programming now can only coincide with a speculative calculus able to introduce incomputable algorithms into actual spatiotemporalities. Here order no longer emerges out of chaos. The emergent properties proposed by the autopoietic self-organization of second-order cybernetics were only the lightning before the striking thunder of chaos: the rumbling noise of incomputable quantities unleashes entropic data at all levels of programming.[35]

As Chaitin hypothesizes, if the program that is used to calculate infinities will no longer be based on finite sets of algorithms but on infinite sets (or Omega complexity), then programmability will become a far cry from the algorithmic optimization of indeterminate processes realized through binary probabilities. Programming will instead turn into the calculation of complexity by complexity, chaos by chaos: an immanent doubling infinity or the infinity of the infinite. If the age of the algorithm has also been defined as the age of complexity, it is because the computational searching for the incomputable space of nondenumerable quantities has become superior to the view that algorithms are simply instructions leading to optimized solutions.

Chaitin's pioneering information theory explains how software programs can include randomness from the start, and indicates that they do not have to be limited to fixed sets of algorithms or to a closed formal axiomatic system. Thus the incompleteness of axiomatic methods does not define the endpoint of computation and its inability to engage with dynamical change, but rather its starting point, from which new axioms, codes, and algorithms become actual spatiotemporalities. Algorithmic architecture, it is argued here, can be conceived precisely as the programming of infinitely random data possessed of volume, depth, and length, which thus come to define actual spatiotemporalities of data. The next section will discuss how these algorithmic spaces have been overlooked in computational and interactive architecture.

1.0.3 Anticipatory architecture

For instance, Brandon Williams/Studio Rocker have argued that algorithms do not give us representations of spatial experience, but are computational

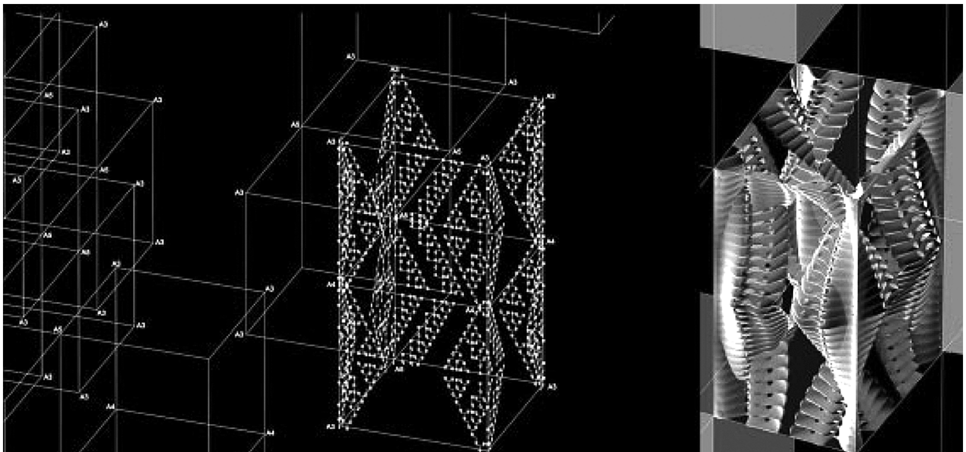

Figure 1.1
Brandon Williams/Studio Rocker, *Expression of Code*, 2004.

processes that can be used to generate unseen structures and unlived events.[36] In *Expression of Code* (2004), Ingeborg M. Rocker argues that coding is intrinsically mutable, as algorithms are malleable parts of information that can become a straight wall in one instance and a twisting column in another. Algorithmic mutations work to fundamentally challenge basic intuitions about what form and space can be. Rocker believes that as changing codes have always characterized architecture throughout history, so the development of calculus into computation can ultimately recode architecture: abstract codes can realize surfaces and structures of another kind in which the incompleteness of form is revealed.[37] It is precisely this incomplete nature of spatial form, I argue, that has become entangled in the incompleteness of computation. Incomputable algorithms have replaced patterns of probabilities (e.g., the forecasting of future scenarios through finite sets, or the modeling of culture through past data) with a metamodel of infinite programmability, whereby random (incompressible) data are included in computation step by step through the continual addition of new axioms.

From this standpoint, not only computational design per se but programming cultures in general have become veritable operators of Omega ciphers: discrete yet infinite states at the limit point of algorithmic computation. These discrete unities of incomputability not only determine the limit point of sequential algorithms but also, significantly, interrupt the digital processing of data into binary states. To put it another way, incomputable algorithms are indivisible yet infinite real numbers that demarcate the limit of computation by defining a discrete yet uncountable space. They atomize the linear continuity of sequential programming. This is, however, another kind of atomization. Not the binary division of zeroes and ones, but the infinite division of potentiality: unity as discrete infinity. If algorithmic computation defines the general matrix of postcybernetic culture, it is because the indeterminateness of programming has become ubiquitous. The operations of programming do not need to be interrupted by an external agent—the environment—in order to expose changes in coding. The continual processing of information is internally and ceaselessly interrupted by incomputable data. This interruption is the space in which the program becomes generative of unprogrammed states, cutting itself off from the continuity of procedural rules.

This is a far cry from metacomputational logic, according to which complexity and variations are the result of the evolution of simple rules. According to this logic, algorithmic mutations explain patterns of emerging complexity, but, as will be argued later in this chapter, these mutations cannot account for patternless data. Against the metacomputational view of a universe contained in simpler axioms, I will argue that incomputable limits are truly intrinsic to computation. This means that ontological complexity or chaotic incompleteness does not emerge from order, but is rather the unconditional condition, part and parcel, of procedural calculations. The appearance of incomputable algorithms (or real infinite numbers) in programming reveals how indeterminate quantities now govern the logic of computation. Incomputable algorithms are not exceptional probabilities, marking, for instance, the moment at which programming breaks down. On the contrary, incomputable probabilities are known probabilities that point toward a new conception of rule. The latter is exposed to a certain indeterminate quantity that cannot be compressed in a smaller cipher or simpler axioms than the output achieved. This new quantitative level of uncertainty is at the core of the metamodeling of everyday operations of programming, designing, measuring, and calculating probabilities through digital, biodigital, and nanobiological machines.

From this standpoint, I will argue against the idea that algorithmic architecture is underpinned by a metacomputational logic through which all can be calculated, but I will also question the biophysical grounds of computation by interaction. If computational architecture and design have reduced the experience of space to the aesthetic form of coding, interactivity has reduced this experience to enactive feedbacks, grounding algorithmic rules in the reality of what is physically constructed. Interactive architecture, it is suggested here, has been too quick to substitute tangible living systems for mathematical forms. Whereas computational aesthetics has reduced experience to the grammar of codes, phenomenal aesthesis has dissolved the discreteness of mathematical sets into a plurality of points of view joined together by one encompassing perspective. The incomputable limits of algorithmic computation instead show that any coding procedure is intrinsically attached to its incalculable quantities, and that aesthetics may imply how algorithmic actualities are infected with an infinite variety of infinities. As may become clearer later, I will propose that algorithmic architecture needs to be explained through another kind of aesthetics, relying neither on the beauty of simpler axioms nor on the continual variation of biophysical interactions. On the contrary, algorithmic architecture is important because it offers us an opportunity to discuss another species of actualities: algorithmic objects, the data structures of which now constitute the immanent data of experiences that do not stem from the directly lived.

Consider for instance the computational programming of a wall, a model of spatiotemporal experience par excellence, which includes the combination of digital and physical data: algorithmic rules, interactive possibilities that these rules can and cannot entertain through physical actuators, the biological data derived from a neoplasmatic design[38] of the wall's materials, and so on. The experience of a computational wall is therefore neither simply programmed into a set of probabilities nor simply left open to the interactive devices that allow rules to become responsive to physical data and direct perception. On the contrary, this experience is involved in incomputable states (algorithmic infinities in sequential calculation) that undermine the seamless linear causality linking algorithmic

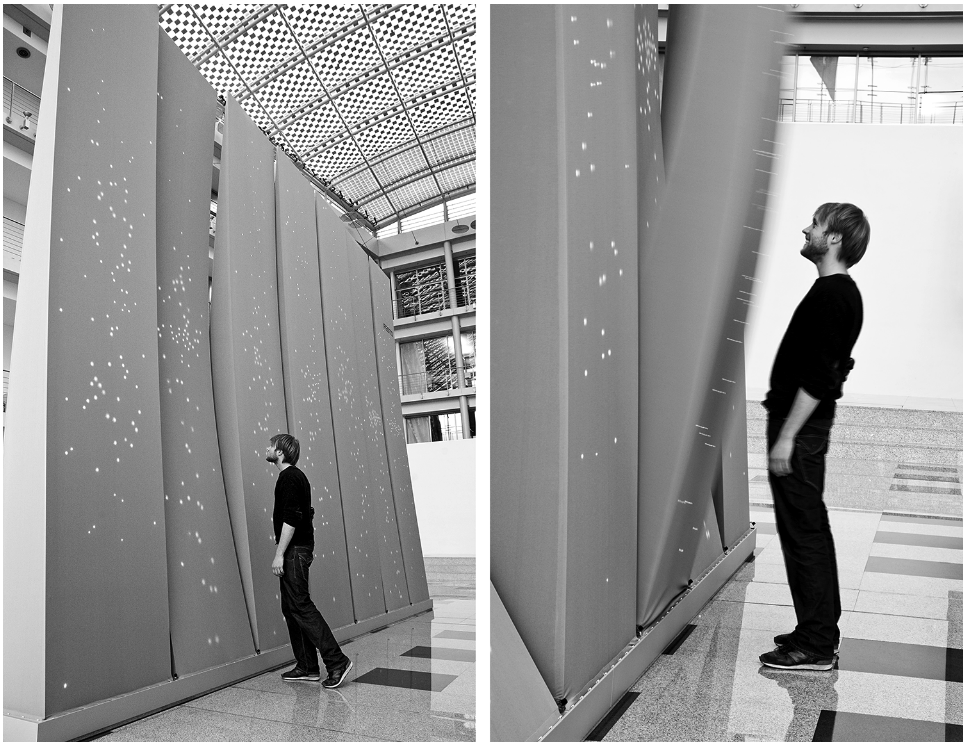

Figure 1.2
Mark David Hosale and Chris Kievid of Hyperbody, *InteractiveWall*, commissioned by Festo (Hannover Messe), 2009. Courtesy of Festo AG & Co. KG. Photographs by Walter Fogel.

procedures, interactive responses, preengineered functions, and the material composition of the wall. These states of incomputability correspond to those data that immediately cut, atomize, and divide information into mathematical calculation, biological responsiveness, and nanodesign of materials.

Let's take for example the *InteractiveWall* project (2009), designed by the architectural group Hyperbody in collaboration with Festo bionic learning networks.[39] The algorithmic programming of this spatiotemporal architectural element, the wall, is carried out by the biophysical interaction with a set of its finite uses. The Hyperbody group wanted to demonstrate that interaction implies not simply a reaction to algorithmic rules, but a transformation of the behavior of people through motion, light, and music in *real time*. In other words, the group conceived of the algorithmic wall as a spatiotemporal zone of experiential transformation, where algorithmic functions could become complicated by the wall's interactive capacity to respond to external stimuli. Despite the Hyperbody group's intentions, however, while the *InteractiveWall* may to some extent succeed in taming spatiotemporal experience, it does so at most by gathering data resulting from the interaction between programmed algorithms and biophysical inputs.

The *InteractiveWall* is made of seven individual panels—each 1.09 meters wide, 0.53 meters deep, and 5.30 meters high—and its structural design is inspired by the anatomy of a fish's tail fin. The wall's panels are programmed to move laterally, with the movement deflected through two electronic drive units. The structure's skeleton moves on its central axis, away from and toward the user, so as to form a convex "hunchback" or a concave "hollow." The wall is programmed to sense the user's slightest movement through ultrasound sensors, which collect data that are inputted back into the computational parameters of the wall's behavior. As the wall moves from one side to another it also deploys changes in patterns of light: an assemblage of 24 circuits, equipped with 20 LEDs, forms a reactive interface between the LED skin and the user. The wall glows brighter when users get closer, and dimmer when they move away. More exactly, the LED skin pulses rapidly and slowly in relation to the position of other wall elements, which are individually programmed so that each has its own region of interactivity while communicating with all the others. Sound adds another element of interaction. The interactive architecture of the wall brings together synchronic movements of calmer sounds and asynchronic motion intended to build an intensified sonic ambience. Each node of the wall is conceived as a member of a choir that sings complex patterns of oscillating chords. The Hyperbody group conceived interactivity in terms of responses that are not algorithmically programmed, since programming is tailored here to respond directly to the user's perceptions. In order to achieve this interaction between algorithms and biophysical inputs, the group used Festo-designed electric drive units that could convert signals from ultrasound sensors into motion so that the wall could, for instance, shift away from approaching visitors.

Following the *InteractiveWall* project, also in 2009, the Hyperbody group designed the *Emotive Wall*, a new prototype, which was intended to highlight more clearly than its predecessor the existence of a mutual, dynamic interaction between programmed rules and user responses. In contrast with the *InteractiveWall*, which was designed to respond to the user, the *Emotive Wall* was more specifically designed to develop the computational persona of the programmed wall. In the words of its designers, this is "a wall that can move because it wants to."[40] This architectural object is thus no longer conceived as delimiting the interaction between mathematical probabilities and biophysical contingencies, but itself exposes its algorithmic and biophysical layers of composition, circumscribing its reality as an actual object.

By designing algorithmic models that are open-ended and attuned to biophysical contingencies, the Hyperbody group also strictly followed the science of emergence and spontaneous order, according to which complex patterns derive from simple rules, in order to model the behavioral patterns of the wall. Here the complexity of the wall was mostly based on the interactions of localized contingencies, such as the specific responses of flashing tails. In other words, local biophysical inputs were devised so as to add indetermination to the whole programmed structure of the wall. Rather than relying on the external input of visitors to create movement and to generate complex behavior, the *Emotive Wall* is now designed to respond to its own biophysical changes. Local variations of data input thus affect the whole behavior of the structure by inducing or stopping movement.

For this project, the Hyperbody group has specifically used concepts of swarm architecture to highlight how computational programming implies a new relation between space and matter, as these both become generated by codes. It could be argued therefore that the *Emotive Wall* prototype may be going a step further compared to those interactive systems whose behavioral patterns are derived from fixed rules. The dynamic configuration of computational architecture that can be found here is not a result of reactive responses to algorithmic programming. On the contrary, the computational architecture of the wall is understood as an *anticipatory system* that is apt not only to program responses, but also to generate potential conditions for interaction from the open-ended evolution of algorithms. With such devices as electric and pneumatic drive units combined with software programs of control, regulation technologies, and sensor systems, this algorithmic programming is transformed into an anticipatory architecture.[41] What is anticipated here are probabilities of interaction, which are not the indeterminate probabilities that were previously defined as incomputable. The *Emotive Wall* is instead an instance of algorithmically adaptive space imbued with interacting components. It aims to provoke anticipation rather than reaction, forcing both software and interactive devices to foresee conditions in which spatiotemporal change can be experienced. Nevertheless, precisely what is missing from these interactive projects is an engagement with the algorithmic nature of spatiotemporal experience.

In particular, swarming models are used here as a means of generating complex algorithmic behaviors, and to explain spatiotemporal experience in terms of a self-organizing autopoietic system that evolves in time through interaction with the environment. Biophysical interactions are therefore the enactors of algorithmic change. From this standpoint, the *Emotive Wall* does not fully work as an example of anticipatory architecture, because here the spatiotemporal experience is derived from physical data, which works as an input for the entire algorithmic architecture. As such, the *Emotive Wall* overlooks two main problems. On the one hand, these experiments in interactive architecture do not acknowledge that algorithms are actual entities, not simply a simulation of physical data. These entities, as discussed later in the chapter, are actuals without being biological and thus need to be addressed according to their own spatiotemporal structure. On the other hand, Hyperbody focuses on the interaction of/with biophysical data in order to explain change.

Recently, algorithmic architectures have engaged with another kind of data collected from materials designed at the atomic and nano levels. In particular, the nanofabrication of biomaterials that can program, control, and sustain cellular structures that grow, evolve, and mutate are becoming central premises for the development of nanoarchitectures.[42] Here the algorithmic programming of spatiotemporalities has entered the space of atomic structures[43] so as to grow systems that anticipate the conditions of possible responses. For instance, Anders Christiansen's nanoarchitectural design of a *Homeostatic Membrane*[44] shows how the intersection of artificially designed molecules creates a responsive interface between the interior and the exterior walls of cells, anticipating the probability of uncertainty in sequential processing.

As opposed to interactive architecture, according to which spatiotemporal experience is defined by a change in the system induced by biophysical data, nanoarchitectures are spatiotemporal structures of anticipation characterized by incomputable data, corresponding neither to mathematical nor to physical inputs. From *ubiquitous computing* to the nanofabrication of walls, smart objects, and clothes that sense and anticipate (or productively prerespond to) changes in atmospheric pressures, moods, sounds, images, colors, and movements, incomputable data have infected the general ecology of media systems.

As predicted by Mark Weiser's dream of an age of ubiquitous and calm technology, all post-desk digital machines are now embedded with a *seamful* environment of data, in which each level of computation extends into another: not through seamless compatibility but through the incorporation of incomputable data within systems.[45] If digital computation has come to characterize the invisible architecture of everyday space, the pervasive extension of algorithmic logic has now become attuned to alien regions of perception and cognition. These are zones occupied by abstract yet real incomputable states that interfere with computational calculus by anticipating new conditions of spatiotemporal experience. However, in order to appreciate the nuances of anticipatory architecture further, it is important to look closer at, and thus to distinguish this architecture from, the notions of interaction and responsiveness that have been developed in the context of ubiquitous computing.

1.1 Background media

The history of old media technologies comes to an end when machines not only handle the transmission of addresses and data storage, but are also able, via mathematical algorithms, to control the processing of commands.[46]

As Kittler reminds us, the introduction of mathematical algorithms into machines turned media into processing systems of command, in which interaction was just the result of feedback operations of control. With algorithmic machines, then, the system became extended to include the user, now part and parcel of an infinite series of loops, in which all forms of input were equivalent to one another.

The algorithmic age therefore also corresponds to the now realized dream of the post-desktop culture of ubiquitous computing.[47] According to Lev Manovich, the post-desk invention of cultural computing was not simply determined by the rise of the personal computer industry, the adoption of graphical user interfaces (GUIs), the expansion of computer networks, and the World Wide Web, but is to be found in the revolutionary ideas that transformed the computer into a "metamedium."[48]

Similarly to interactive architecture, the ergonomic design[49] of interactive media has left behind the algorithmic "stuff" of computation by burying information processing in the background of perception and embedding it deep within objects.[50] In other words, with interactive media, information processing has become transparent. This also means that the postcybernetic realm of interaction requires no direct response or execution of instruction. Instead, and as in the case of Hyperbody's *Emotive Wall*, interaction now includes a computational tendency to anticipate responses and programming indeterminacies that stems from within the system, and need not rely on direct sense perception and cognition. These new epochal traits of postcybernetic control coincide with the arrival of ubiquitous computing: with the withdrawal of mediatic action to the background of perception and direct experience.

Projects such as Hyperbody's *InteractiveWall* extend Mark Weiser's vision of the "age of calm technology,"[51] in which the design of computational space involves the programming of architectural networks. Small objects such as mobile telephones, Blackberries, iPods/iPads, digital cameras, radio frequency identification tags, GPS, and interactive whiteboards are ceaselessly mapping—deterritorializing and reterritorializing—this background architecture of invisible algorithms. This kind of architecture coincides not with the cyberspace of data simulating physical conditions, or with social networks of instantaneous communication that are determined by the activities of the users. Instead, the realized age of calm technology announces the now diffused conditions of cultural programmability: here algorithmic computations are entangled with operating systems, search engines, databank structures, miniaturized hardware pieces, molecular growth and adaptation, nanodesign of atoms, randomization of percepts and affects, the nonsensuous rendering of coding, decoding, and recoding processes, layers of database incorporation, annexation, and expression.

While ubiquitous computing announces the deep burial of algorithmic processing in ergonomically designed objects of interaction, the reality of infinite quantities of data can no longer be contained in the axioms of universal calculus. From this standpoint, all forms of cultural programmability reveal that each and any step of programming is hosting incomputable states, able to interrupt the ubiquitous continuum between computational objects (mobiles, iPads, and social media in general).

Weiser's pioneering work in the field of mediatic applications of computing already sees computer objects not as isolated objects placed on desks, but as *ambient* objects that surround us everywhere: walls can be turned into electronic boards and displays, books become electronic information stores, and cameras act as digital picture libraries.[52] According to Weiser, ubiquitous computing changes not only the locations of digital machines but also the use of such machines. The human user no longer activates computation, but is now incorporated in the programming system, as she or he can now indirectly profit from the computational capacities hidden in mundane objects.

This stealthy intrusion of algorithmic programmability into distinct ordinary objects is also symptomatic of a new twist in the long history of ergonomics, the science that aims to optimize the interactions among humans and other elements of a system by fitting perceptual, motor, and cognitive capacities into the latter. The human user is no longer the operator of an inflexible machine of calculation, but has become a component or trigger of sequential operations. It is therefore possible to suggest that the neoergonomic character of postcybernetic computation works not to optimize, but more generally to anticipate the probability of indeterminacy in all algorithmic architectures.

Second-order cybernetics insisted on the role of the observer, who was able to mediate given instructions and to change the rules of objects. The new focus on the central action of the observer led to the development of user-centered interface and to the design of computers as portable media. Symptoms of this development can be found not only in Weiser's ideas but also in the work of another active researcher working at PARC labs during the same years: Alan Kay. He was the pioneer of object-oriented programming and devised the first object-determined language for computers, called Smalltalk. In particular, users developed this software thanks to Kay's vision of a programming language organized around objects rather than actions, data rather than logic.[53]

For Kay, everything was an object: a biological cell, the individual computer on a network, a black box. A computational object, he insisted, was composed of code or sequences of algorithmic instructions through which it received and sent messages. According to Kay, object-oriented software could not therefore distinguish between data (or structures) and code (functions), as both data and code were merged into an undividable thing:

the computational object. The user of the object, therefore, did not need to see what was inside the back box in order to receive and transmit messages: the user could develop an intuitive approach to computation rather than having to become a computer programmer.

In Kay's opinion, ensuring the smooth running of software did not require opening the black box.[54] His programming language composed of objects was thus ready to be manipulated and changed by users in a creative and nondetermined way. Far from being derived from logical calculations that need to be executed, this language offered a view of computation that was open to the interaction of programmed/able objects. According to Manovich, this shift toward object-oriented programming transformed the computer "from being a culturally invisible technology to being the new engine of culture."[55]

Just as Kay's designs for portable computers in the 1970s anticipated the portable media transformations of the late 1990s, so too can the object-oriented software of Smalltalk be seen to pioneer the cooperative development of open-source. This dates back to 1995, when Kay launched Squeak: an open-source software that required users to participate in its programming evolution by adding and expanding (and inventing) computation, rather than by revealing already programmed sources.

As Manovich points out, since the '70s Kay's vision has provided users with a programming environment and already-written general tools for the invention of programming languages. By the '90s, in fact, Kay was devising programming platforms that enabled users to directly use software objects, thus opening software to user interaction. This form of object-oriented software interaction was conceived as an alternative mode of programming, and was specifically influenced by the notion of computation associated with second-order cybernetics. Here the users' aggregation of software objects could grant the emergence of novelty in interactive media systems.

But the development of object-oriented programming and interactive media in the '70s also led to a transformation of computational logic. According to Kittler, this logic involved strategies, developed during World War II, of cybernetic command and control over information. In particular, Kittler explained that Turing had proved that the computerized calculation of recursive functions ultimately exhausted the whole domain of human computability. In other words, computers as finite state machines, which Turing deemed to be predictable from start to end, followed not the laws of nature but rather "the very logic of decision-making strategy, and information war."[56] This is why the human postal system, for example, came to an end.

According to Kittler, "data, addresses, and instructions, can handle each other by means of digital feedbacks such as if-then conditions and programmed loops."[57] Thus, the principle of automated computation presupposed that input data were ceaselessly transformed into finite numbers of discrete states able to register, transmit, and calculate any data without human intervention. From this standpoint, as Kittler pointed out, old media became devoured by the universal medium of computation, which excluded the redundant noise, the unreliable contingencies of human perception and cognition from the start.[58] If the origin of digital computers is rooted in strategies of control, then this, as Norbert Wiener explains, is because taking the humans out of the decision-making loops entailed that the job of prediction could become more effective (e.g., the trajectory for hitting a moving target, such as a plane, could be more closely approximated).[59]

The creation of artificially intelligent machines, in which finite sets of rules could process vast and complex amounts of information, was central to first-order cybernetic research, which focused on programming inputs rather than on using algorithms to extend capacities of interaction and communication between systems. But the shift away from automated calculation to personal computing through object-oriented (interactive) software inevitably extended the logic of computation onto social systems.[60] In brief, the shift toward second-order cybernetics implied the extension of computation into social modes of organization through the user-friendly transformation of software languages. At the core of this extension is a double-edged sword of control and freedom that is entangled with open-source models of algorithmic architecture.[61]

From this standpoint, Mark Weiser's view of ubiquitous computing can be viewed alongside Alan Kay's reformulation of media objects as invisible, user-sensitive, semi-intelligent, knowledge-based electronics and software, which were thought to be able to merge with human, individual, biological brains. Thus, while Kay added interactivity to algorithmic computation, Weiser's visions of ubiquitous computing extended interaction to automated machines and not just to users, thus devising a universal media-machine able to encompass a networked architecture of immediate communication and the autoregulation of data systems. Ubiquitous computing therefore now includes the design of artificial intelligence architectures that are able to respond to external action and to smartly correlate distinct networked systems of information. Ubiquitous media mix the computational logic of the universal computing machine with the interactive, adaptive, and open learning systems of user-friendly communication.

Weiser's idea of calm technology turned computers into everyday objects that no longer had to rely on the attention of users in order to work. On the contrary, as media objects, computers had to recede into the background so as to become ambient, and now operate in a manner that involves peripheral modes of perception. Digital media have been described as forming the age of the "information bomb," an age characterized by an everyday state of overstimulation,[62] in which technology constantly demands attention. Weiser's conception of a calm and comforting technology, however, already predicted postcybernetic ubiquitous media's tendency to push perception and cognition into the background architecture of algorithmic processing.

According to Weiser, calm computation activated peripheral zones of thought and feeling and allowed latent parts of the brain to become differentiated capacities. What receded into the information ambience was, according to Weiser, not an aimless, blank noise, but rather highly differentiated potentials that afforded a detailed set of actions. Weiser used the term "locatedness" to refer to the capacity of the peripheral area of the brain to instill a familiar feeling of connection with the world around it. He insisted that the operative functioning of the peripheral brain was crucial to the feeling of calmness in the sea of information processing.

Weiser quoted three examples of calm technology: inner-office windows, Internet Multicast, and artist Natalie Jeremijenko's work *Dangling String* (also known as *Live Wire*), which was made between 1995 and 1999.[63] In particular, this installation was used to explain how the formation of a new center of attention—the movement of the dangling wire that was proportional to the number of packets on the network—revealed the background behavior of the network traffic. This dynamic behavior of the wire came to coincide with an "intuitive peripheral representation of the network activity" (i.e., the increased traffic on the local area network was evinced by a higher frequency of wiggles). As opposed to the visual representation of network traffic based on symbols that demanded our interpretation and attention, Weiser argued that Jeremijenko's installation demonstrated that we become more attuned to information if we attend less to technology. In particular, Weiser associated the physical movement of the string with the brain's peripheral nerve center, since the *Live Wire* consisted of an installation of an eight-foot piece of plastic spaghetti hanging from a small electric motor mounted in the ceiling. The motor was electrically connected to a nearby Ethernet cable, so that each bit of information caused a tiny twitch. A very busy network would then cause a whirling string with a characteristic noise, while a quiet network only involved a small twitch every few seconds. As the string could be placed in an unused corner of a hallway, it blended into the environment, remaining visible and audible to many offices without being obtrusive. For Weiser, the *Live Wire* was an instance of how algorithmic computation could become the ambient technology of social modalities of interactions.

Calm technology no longer required direct perception and operation, but became ergonomically attuned to the peripheral regions of perception and cognition. With ubiquitous computing, the algorithmic background is fully realized through remote ambience programming and through the deep burial of rules. Ubiquitous computation then marks the advance of an entirely new algorithmic architecture that relies on an ever-differentiating background, which becomes the potential field of foreground operations.

But what does this new movement of foreground and background computation actually imply? Does the algorithmic background of peripheral perception stand for what remains invisible to perception and cognition altogether? From this standpoint, one may wonder whether this movement is (yet again) to be understood as the metaphysical correlation between the visible and the invisible, between the veiled and the unveiled. Nevertheless, it is possible to argue that the increasing complexity of background operations in forms of ubiquitous computation that have fully incorporated users into its algorithmic processing, thereby establishing a movement between foreground and background, may not simply take place between the perceiver and information, or between the center and the peripheral capacities of cognition and perception. Instead, this algorithmic background may perhaps be explained in terms of immanence, whereby incomputable quantities of thought and affect infect computable procedures. In other words, what if ubiquitous computing is not simply the new incarnation of formal intelligence in physical machines (which interact among themselves without the direct attendance of a user), but is overshadowed by random algorithms transforming the spatiotemporality of experience?

In postcybernetic culture, the tension between a metacomputational world made of discrete objects and an autopoietic world of generative codes open to users' interactions has led to a reconceptualization of computational models themselves, turning codes into enactive and responsive agents able to be modified by the activities of the environment. Inasmuch as Alan Kay devised an object-oriented software in which the black box of finite algorithms did not have to be directly operated by the user, he embraced an autopoietic conception of computation by which the code could respond to the user and vice versa. To put it simply, Kay already foresaw that algorithms had become background objects that could adapt, change, and evolve.

Ubiquitous computation is therefore a generalized extension of this self-organized ambience that comprises the most ordinary objects of communication. In particular, ubiquitous computing aims at incorporating biophysical contingencies, or those unprogrammable situations that users, participants, or environmental factors can make available to programming by adding more variables to intelligent networked devices. Nevertheless, this neoergonomic tendency to preadapt the computational system to contingent changes, and to move algorithmic functions into peripheral zones of attention such as non-direct cognition and emotion, has not merely marked the transition from a "programmed" to a "programmable" object. More importantly, this transition has meant that algorithmic architecture has become an anticipatory metamodeling of incomputable data that cannot be contained in mathematical forms or physical objects.

For instance, as the algorithmic architecture of the Hyperbody group suggests, the incorporation of biophysical contingencies into an open-ended programming requires an anticipatory conception of space. Kas Oosterhuis insists that interactive architecture must not be concerned with designing buildings that are responsive and adaptive, but must be proactive and propositional, able to anticipate new building configurations and actions in real time. This means that complex adaptive systems are designed to impose a social behavior on building materials, which are now programmable actuators. According to Oosterhuis, it is more important to design the relation between these programmable materials than the relation between the building and its inhabitants, since it is the former and not the latter that makes the building an active environment. As an instance of ubiquitous computation, Hyperbody's projects conceive the building as a self-organizing, interactive entity.

This notion of interactivity, however, despite being rooted in the biophysical and evolutionary vision of autopoiesis, seems to be a far cry from the more general notion of responsive environments in which the human user seems to directly animate algorithmic objects. Here the solipsistic ontology of autopoiesis defines a center able to incorporate interactive parts into one architectural whole. It seems that the recent trend toward "responsive environment" projects seems to be exactly framed according to this ontology.[64] As Lucy Bullivant explains, the notion of interaction has been challenged because it merely describes a unidirectional pattern of communication in which software delimits the potentialities of users.[65] The notion of responsiveness, on the other hand, includes visitors, participants, and users: the ultimate manipulators of structures, buildings, and spaces. In particular, responsive environments define "spaces that interact with the people who use them."[66] Responsive environments are said to have phenomenological impact, "meaning that the body is able to directly experience its environment in a very direct and personal way."[67] By focusing on how the output of the observer/participant is able to reconfigure the relationship between the input and the output, responsive environment projects, such as the sound installation *Volume* by UVA, do not require people to understand the algorithmic model in order to become part of the overall architecture.

In *Volume*, visitors can indeed change the volume and arrangements of the sonic environment through their movement around the installation. It is this physical movement that animates the algorithmic set of instructions of this architecture, which comprises a grid of LED lights structured by 46 2.5-meter columns which form a sound orchestra.[68] Similarly, architect Lars Spuybroek of NOX and artist Q. S. Serafijn's project *D-Tower* most clearly explains how the motor of interaction is placed in the hands of users or participants. The *D-Tower* project allows the users to remotely confer emotional states on the physical object through a website questionnaire, the results of which are turned into algorithmic instructions that animate the tower in the form of different colors. According to the designers, these colors give us an insight into the different moods of the city.[69]

Although Bullivant sets notions of interaction apart from responsiveness, responsive architecture clearly aims to replace algorithmic design with an environment of users. Both conceptions comply with the autopoietic view of an adaptable system that is able to change. From this standpoint, it is possible to conclude that the distinction between interactivity and responsiveness is only marginal compared to the ontological weight that the idea of a self-organizing evolutionary system has imposed upon conceptions of spatiotemporal experience.

For instance, the Oosterhius office ONL's *Digital Pavilion* in Korea, located in the Digital Media City in the Sangam-dong district of Seoul, consists of a series of interactive installations that compose a parametric morphology based upon a 3D Voronoi algorithm,[70] and thus embraces "ubiquitous computing at its full potential."[71] The *Digital Pavilion* is therefore composed of Voronoi cell structures that are equipped with built-in actuators able to alter the length of the cell beam in real time. Users also embed their personal details into a 4G/WiBro in order to control these actuators. A search engine, which reconfigures the users' media content, and which thus builds proactive profiles for the visitors that are uploaded on online databases, processes these details. RFID tracking of individual users is also fed back into the system's algorithmic architecture to generate real-time profiles over the Internet about the potential interaction between people as they visit the *Digital Pavilion*. But the visitors are not simply passive entities incorporated into the algorithmic architecture of the structure. The ubiquitous computation of interactive devices was designed to allow users to engage in four types of socially interactive experiences that trigger alterations to the *Digital Pavilion*'s building structure. On one of the floors of the installation, the hard kinetic pneumatic structure becomes a soft organic architecture composed of a point cloud of tens of thousands of programmable LEDs, which vary in densities and create a spectrum of high- and low-resolution effects for the visitors, who can have gameplay experiences that range from action/shoot-em-up games to social chat games, and so on. This interactive animated building is intersected by another level of interaction through online multiplayer games and urban games based on GPS, GIS, RFID, and wireless technologies, which extends the polygonal architecture based on the distance between discrete points into another environment.

The resultant kaleidoscopic experience of this media-rich, translucent, and fully immersive space that foregrounds the activities of interactive algorithms seems to be defined by probabilistic patterns that rely on programmed inputs and responsive behavior. But this project has nothing of the anticipatory architecture that seeks to include incomputable probabilities into the system of seamless interaction and responsive participation. The problem with an algorithmic architecture composed of interactive parts, whether these are human or nonhuman, or are fed by sensorimotor data derived from visitors or other nonliving sources and collected by actuators, remains precisely embedded in the idea that the bidirectional communication of many objects leads to the emergence of the architectural whole. This is a problem inherent to interactive architectures in which the algorithmic background is determined by cellular automata: self-organizing units of computation that self-change through time, and yet remain finite sets of instructions, always already reducing the possibility of a new spatial experience to closed sets of probabilities. This is why the kaleidoscopic spatiality of the *Digital Pavilion* only ever remains an already-lived experience, a set scenario.

Far from anticipating the possibility of a novel experience of space, interactive and responsive environments seem to remain embedded in the digital paradigms of metacomputation: grounded, that is, in the autopoietic ontology of self-organization by environmental adaptation. Here architecture is a system of variable codes responding to variable inputs. What is left behind, however, is the possibility that algorithmic architecture may be describing algorithmic objects, and that these objects do not simply coincide with finite axioms. In other words, autopoietic ontology overlooks the possibility that indeterminate change could also stem from within computation, and constitutes a problem for computation before it becomes an issue of external interaction between object and environment. To put it simply, the autopoietic view at the base of interactive and responsive architecture misses the point that the environment—which is understood here as an extra space of incomputable data—is within the algorithmic object. This is an important point to grasp, and in order to explain it more fully we will have to engage with theories of the object and of actuality that do not ontologically differentiate between human and nonhuman, between animate and inanimate entities.

Before discussing these theories, however, I will first outline the debate that underpins the theory of metacomputation, and thereby provide a stronger argument against the idea that universals or set codes resolve all forms of complexity. This discussion will explain why the metacomputational model has failed to describe the actuality of algorithmic objects, and has instead confined the indeterminacy of software to the safe ground of mathematical axiomatics. It is this safe ground, I will argue, that reappears yet again in algorithmic architecture. Although it is often accused of being too abstract, in the cases discussed below algorithmic architecture has *not been abstract enough* to reveal how algorithmic objects are the spatiotemporal matrix of the present.

1.2 Metadigital fallacy

The ontological view of metacomputation has characterized those strands of algorithmic architecture that have been more closely concerned with the degree to which computation provides an entry point into the digital material of design. An example of this can be found in the *Milgo Experiment*, also known as the *AlgoRhythms Project*, which was devised by architect-morphologist Haresh Lalvani. Since 1997 Lalvani has been working with Milgo/Bufkin, a metafabrication company, to realize curved sheet-metal surfaces designed through digital programming. All his experiments are

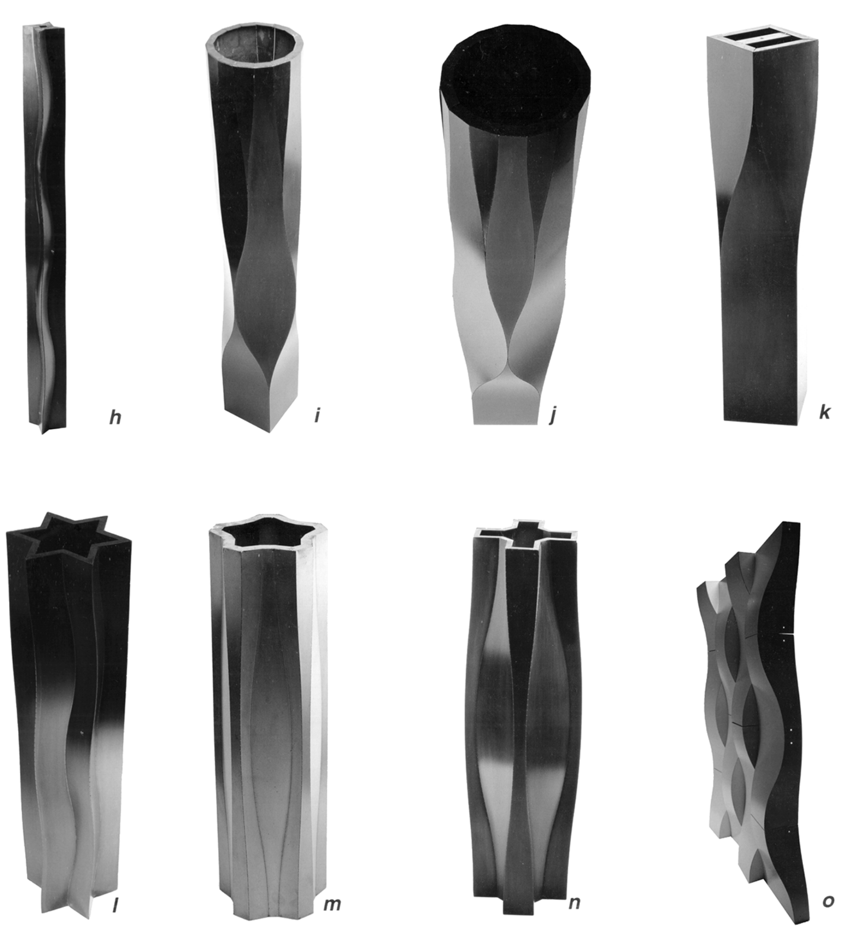

Figure 1.3
Haresh Lalvani, *Prototypes of Columns and Surfaces in Sheet Metal with the Company Milgo-Bufkin*, 1997–1999. Courtesy of Haresh Lalvani. Photographs by Robert Wrazen.

devoted to the development of the *Morphological Genome*, which Lalvani describes as the search for a "universal code for mapping and manipulating any form, man-made or natural."[72] In his projects, morphology (or the process of shaping) and making (or the actual fabrication of the form) are not separate, but rather become merged into a seamless whole in which form and matter become one.

These projects show how curved universal forms, and not straight lines or flat planes, can be obtained by building rigid curved surfaces made of uncut sheets of metal. These curved structures are developed from a single continuous metal sheet shaped by algorithmically generated geometries. Lalvani's morphological meta-architecture permits endless variations on the same theme, which result in continuously curved columns, walls, and ceilings. Lalvani's central conception of morphological meta-architecture is based on a bottom-up morphological evolution of genes, where form and fabrication are linked from the start. Each gene is a cellular automaton that specifies a family of related parameters, and each parameter is controlled by a single variable of form corresponding to a base in the DNA double-helix genome. The universal structures of form and matter, for Lalvani, must be derived from simple genetic rules.[73] In other words, and in conformance with fundamental computational principles, he believes that the infinity of all possible forms can be specified by a finite number of morph genes. As he affirms: "I am expecting this to be a small number."[74] Computation, however, is used here not to solve already existing problems, but rather as a generative process that aims to liberate calculations from probabilities and to demonstrate that "new morphologies (which lead to new mathematics and new architecture) are possible."[75]

Lalvani's model of a continuously generating morphological genome therefore embraces the predicates of a digital metaphysics, according to which cellular automata and discrete entities are universal codes that can produce, just like DNA, an infinite variety of forms and processes that enable the construction of new material spaces. The programming of forms, therefore, is predicated here on the automorphogenesis of forms, wherein a short computer program—a genetic code for instance—guarantees the autopoiesis (self-making) of the universe.

Lalvani's use of algorithmic architecture is not too far removed from the fundamental theories developed by so-called digital philosophy, according to which digital or discrete codes are the kernel of physical complexity: code is ontology, that is, and finite sets of algorithms are the axioms upon which it is possible to build any complex world. According to digital philosopher Edward Fredkin, all physics can be explained through the simple architecture of cellular automata, or discrete entities that form a regular grid of *cells*, each one of which exists in a finite number of spatiotemporal states.[76] According to this digital view of physics, the universe is a gigantic Turing cellular automaton: a universal machine that can perform any calculation and program any reality through a finite number of steps. While truly Turing-complete machines remain physically impossible to realize, since they require unlimited amounts of data storage, Turing completeness is nonetheless achieved through physical machines or programming languages that *would* be universal *if* they had an indefinitely enlargeable storage. Fredkin argues that particle physics can be explained as an emergent property of cellular automata. In other words, for Fredkin, cellular automata are the ground on which physics can be explained. In short, if cellular automata are the ground then the universe is digital, as it would then be built upon discrete units, which reveal that space-time is not continuous.

From this standpoint, what determines the ultimate digitalization of the universe is the calculation of infinite probabilities, the real possibility to actualize infinity and to design a mathematical language able to turn the potentiality of infinity into sets of axioms. This atomic conception of the universe divides the Parmenidean infinitesimal continuum into finite small particles, or atoms, out of which the complexity of the universe is derived.[77]

Similarly, Stephen Wolfram, physicist and creator of the Mathematica program, postulates the "principle of computational equivalence," according to which all complex behavior can be simulated computationally. Simulations are "computationally irreducible" and do not represent natural behavior, which is instead "generated by computation."[78] Digital computation complies with the metaphysics of discrete mathematical entities moving in empty space, in which the plurality of the universe is reducible to indivisible units, or atoms. By setting up a series of simple programs and running them through computation, cellular automata generate complex or irregular structures.

Wolfram's notion of cellular automata has been widely adopted in algorithmic architecture as a means of exploring irregular or complex geometric forms that stem from simple rules. Wolfram explains that "even though the underlying rules for a system are simple, and even though the system is started from simple initial conditions, the behavior that the system shows can nevertheless be highly complex. I argue that it is this basic phenomenon that is ultimately responsible for most of the complexity we see in nature."[79] Wolfram calls the discovery that simple rules can generate high levels of complexity rule 110. As he points out, however, the story is more complicated. Not only do simple cellular automata, determined by the same rules or by the same finite algorithms, generate complex structures; there also remains no trace of uniform simple cellular automata in the varied complex structure to which they give rise. This means that despite being determined by the same rules, cellular automata do not do the same thing: while some maintain a regular pattern, others become part of other localized structures, developing different configurations of cells, changing, for example, the sequences of colors, and thus producing behaviors that are totally different from those established by the initial rules. Therefore simple programs do not merely generate complex forms, but are also encoded in all forms, processes, and matter in different ways.[80] Yet the view that original codes evolve in unrecognizable ways may not be enough to explain how infinite states are still a problem for computation, no matter how complex the evolutionary journey of a set of codes can be.

While using cellular automata and the binary view of the universe, Lalvani also wonders whether there are infinite states between zeroes and ones, which he is unable "to carry . . . to all levels of form."[81] He concludes by suggesting that discrete and continuous states and everything in between may be coexistent in the universe, and that while physical reality, as Wolfram claims, may have a discrete basis, it may also appear to be perfectly continuous on an experiential level.[82]

Nevertheless, the fallacy of metadigitality does not simply imply that the ground of the universe, as Lalvani observes, is discrete, while physical experience explains the analog continuity between things. Instead, this fallacy stems from believing that cellular automata animate formal processes and matter. Architect Neil Leach, for instance, has argued that the logic of swarm intelligence has challenged the computational methodology that rests on the discrete internal logic of fractals, L-systems, and cellular automata.[83] In particular, he specifies that while fractals and L-systems are not flexible enough to adapt their behavior to external stimuli, cellular automata do interact with their neighbors, yet ultimately remain anchored to a fixed space and are unable to change their underlying grid.[84]

Kwinter reminds us that the age of the algorithm makes space and matter indiscernible. However, this is not because they mimic each other *in* and *through* simple finite rules or cellular automata. No ultimate synthesis, whether metamathematical or metabiological, can act as the universal glue that binds matter and space. The computational view of the universe maintains that the formula for the existence of all matter and space can be contained in simpler programs (and grids of cellular automata) out of which complexity is generated. As opposed to this, it is argued here that algorithmic architecture is not simply a metacomputational field of application: on the contrary, algorithmic design is forced to face the problem of infinity, which is intrinsic to the computational process. For this reason, one must question the idea that cellular automata are the ultimate generators of objects and structures. Similarly, the notion that space and matter have become equivalent through algorithmic processing, or through the repetitive patterns of cellular automata, must also be challenged. Kwinter's insights into algorithmic design therefore seem to end up supporting rather than transforming the metacomputational assumptions that lie at the core of digital philosophy.

My contention here is that metadigitality is marked by a double fallacy. On the one hand, finite sets of algorithms or simpler instructions cannot program, contain, or reduce material complexity. On the other, the adaptation of a generative, evolutionary, biological model of complexity that explains how simple rules—cellular automata—need to be understood as genetic programs is also problematic. While the former approach remains entrapped in a computational model predicated on the idea that software calculation instructs matter, the latter model mainly ends up conflating computation with matter itself. The fallacy therefore consists in reducing algorithms to finite quantities on the one hand, and grounding algorithmic quantities on biological and more specifically genetic models of evolution on the other. What is missing from this picture is a reconceptualization of algorithmic quantities. These quantities, it is suggested here, do not simply coincide with a determinist method of measuring. Instead—and this is important—algorithmic quantities also reveal another face of computation: the deployment of infinite numbers that cannot be computed or reduced to smaller axioms. Far from looking for a simple pattern beyond complexity, I argue here that algorithmic architecture cannot overlook the fact that the limit of computation lies in infinite sets of data. Instead of holding onto a mathematical ontology, and casting it as the ultimate Holy Grail that underpins all systems and structures, algorithmic architecture may instead need to address the incompleteness of axiomatics and rational logic. Yet it is not enough to search for such incompleteness in the analog variations of biophysical inputs, as interactive and responsive design do. Instead, the challenge is to embrace the reality of infinite algorithms, which point instead at a suprarational, immanent, and speculative thought, which cannot be reduced to Being or ultimately fuse the qualities and quantities of objects (matter) and their relations (space).[85]

Computation does not therefore simply correspond to the elimination of qualities from calculation or their formal axiomatization. Algorithmic computation instead describes, but cannot be reduced to, biophysical matter, as it exposes the qualities and quantities of objects at an abstract and yet all too real level. Here qualities are at the same time incomputable quantities, which cannot be summed up in discrete binarism or contained in self-generated wholes. Similarly, quantities do not simply define that which is calculable, but also, and increasingly, reveal the reality of infinity, the precision of indefiniteness.

From this standpoint, the view of universal computation based on the binary rule of probabilities, founded on the Turing machine, cannot explain the incompleteness of any object (whether a physical, digital or biological object). Contrary to Wolfram's metacomputational universe, algorithmic information theorist Gregory Chaitin sustains that the limit of computation is not simply determined by the time of calculation and the memory storage of the Turing machine. This means that, according to Chaitin's theory, the fallacy of metadigitality resides in the fact that digital philosophers understand information quantity in terms of time and space as determined units of measure. Instead, Chaitin argues that information needs to be understood in terms of computational entropy. From this standpoint, even the simplest cellular automata are already infected by complex, incompressible, random information.

As opposed to Wolfram's digital philosophy, Chaitin suggests that physical complexity cannot be reduced to simple rules (for instance DNA understood as algorithmic instructions that generate organisms), or to one formal axiomatic system (subtending universal computation). Similarly, if the metacomputational view of the universe sustains that infinitesimal numbers (used by Leibniz to explain the relations between discrete objects) do not exist, Chaitin argues that infinitesimals cannot be reduced to the integral calculus, which gives us a sum derived from the function of the differential relations between points. Nevertheless, according to Chaitin, there must be a complexity of the discrete number itself, not derived from the ratio between two points.[86] Omega is this anomalous discrete unity, infinite and yet indivisible, a mathematical and yet also a quantum entity.

What can be immediately learned from Chaitin's theory is that not only (analog and biophysical) qualities but also computational quantities are therefore incomplete. In particular, Chaitin's theory suggests that each and any object is at once discrete (and indivisible) and yet composed of infinite uncountable parts. It is precisely this notion of incomplete yet discrete quantities that can transform algorithmic architecture into a metamodel of immanent signs and objects. Similarly, as will be discussed in this book, these incomputable quantities characterize the postcybernetic apparatus of anticipatory architectures, programming data cultures and building data spaces that are immanently experienced but not directly lived.

It is therefore suggested here that algorithmic architecture does not simply reveal the workings of universal codes that generate metamorphic structures of change. Instead, algorithmic architecture can be taken as an example of the actual existence of spatiotemporal data structures infected with incomputable quantities. Far from being a model of existing structures, algorithmic architecture becomes a metamodel of algorithmic objects, which are conceived here as discrete entities imbued with infinity. Chaitin's notion of the incomputable (or discrete infinity) poses a radical challenge to the predicates of first- and second-order cybernetics. In particular, his algorithmic theory questions the assumption that parts (either sets of algorithms or the interaction between algorithms and biophysical inputs) constitute the whole (the mathematical axiom or the autopoietic system). Chaitin's theory demonstrates that parts are irreducible to any totality because they can be bigger (quantitatively incompressible or irresolvable by a simpler solution) than wholes, and can instead overrun them. Algorithmic architecture is not a whole constituted by parts, but rather shows that parts are irreducible inconsistencies divorced from the totality that can be built through them; it works not against but rather with the chaotic parts of information that are comprised neither within mathematical axioms nor within the law of physics. Algorithmic architecture therefore offers us the opportunity to discuss the nature of algorithmic objects beyond pure mathematical and physical models. It thus contributes to the articulation of a new notion of discreteness that may well overturn what is meant by the digital. The next few sections of this chapter will endeavor to clarify what is at stake in this new notion of discreteness.

1.3 Discrete objects

> Within any system, design must take place simultaneously at the level of the object or node, and at the level of the wiring, connection or protocol. Thus, the studio takes the appliance, a discrete object wired for connection in a larger system, as its fundamental design unit.[87]

The Responsive Systems/Appliance Architectures advanced research design studio, at Cornell University, insists on a conception of space that is not an envelope, one that implies a new model of control through the blurring, flattening, and dispersal of hierarchies. In particular, they conceive of appliances as objects or assemblages of objects with embedded intelligence (from kitchen appliances to systems furniture to iPods), and have also devised a generation of new digital objects. These latter are produced through simple kinematic units that are able to combine into larger assemblies. With Alias Waterfront's program, inverse kinematics was used with scripts to model the potential behaviors of the objects. Objects are conceived here as active systems that change over time, and as requiring a method for discovering their behavior and as-yet-unrealized use. Here the object is defined by the interactions generated by its components and by the relations that they have with other objects. The space between these interactive parts, and in which they are themselves located, thus becomes a scenario of potential configurations of objects that are programmed to be partially unplanned.

Nevertheless, according to Neal Leach, the complexity and the dynamic capacities of discretely computed objects—a complexity and a dynamics that define new spatial relations—can only be the result of autonomous design agents which are able to self-organize.[88] For an instance, Leach draws on the Kokkugia network of Australian architects, who have devel-

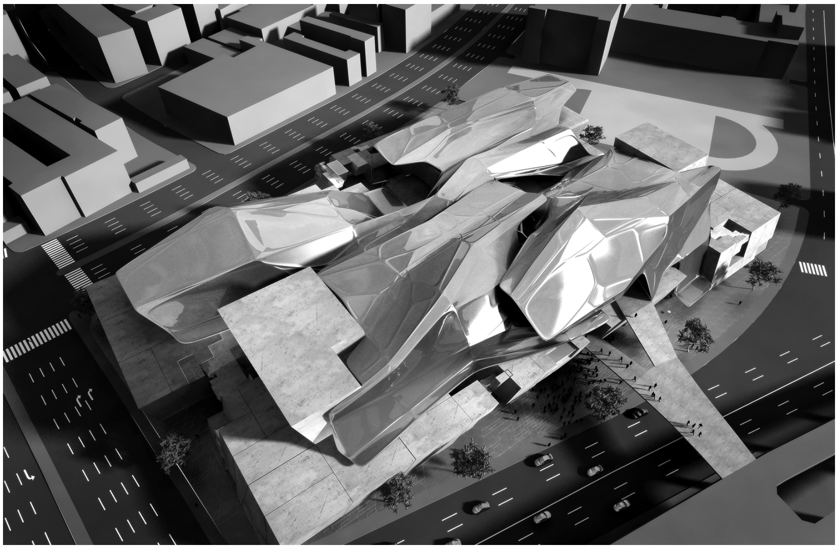

Figure 1.4
Kokkugia, Taipei Performing Arts Center, 2008. Courtesy of Roland Snooks + Robert Stuart-Smith.

oped a multiagent design tool based on the notion of swarm intelligence. This multiagent system does not, however, simply simulate or map actual populations of agents in order to find optimal solutions for urban planning. On the contrary, this is a flexible system based on self-generating objects-agents, which interact with one another and thus reveal spatial mobility.

Leach takes the Kokkugia's design of the Taipei Performing Arts Center (2008) as an example of swarm modeling, whereby interactive self-organizing multiagents define objects in terms of unity and the parts thereof. Here the object is conceived as being both one and many. From this standpoint, the software program used to design the roof of the Taipei Arts Center demonstrates that discrete objects are made of partial relations. This is a network of semiautonomous software agents that reform their original topology and geometrical form by visibly changing some parts of the roof, while others keep their original geometry determined by infrastructural connections. Here the parts of an object are conceived as semiautonomous agents able to evolve their own set of interactions with other objects without reproducing the same set of instructions. Similarly, changes are only dictated by the emergence of contingent solutions. This is why, according to Leach, the space engendered by the movement of these parts is itself a dynamic space defined by multiple discrete objects. From this standpoint, the self-organization of semiautonomous agents defines each and any object of computation as a complex of parts and their interactions.

Nevertheless, this emphasis on self-organizing agents and partially interacting objects seems to be (yet again) proposing the autopoietic fusion of space and matter into one system of interaction. Here space and matter are co-constituted by interactive parts that coevolve over time and deploy their changing morphologies. Therefore the ontological premises of swarming agents turn space and matter into the emergent and not preprogrammed properties of algorithmic objects. These premises do, however, lead to an organic notion of totality in which the algorithmic object is mainly the result of interactive, evolving parts that form a continuously changing whole. This ontological assumption cannot help but reify the notion of a computational continuum,[89] according to which infinitesimal relations between parts make it impossible to discern one object from another. The Taipei Performing Arts Center's roof thus remains the product of the interactions of multiple agents, and is the ultimate effect of their activities. Ultimately, swarm models rely on the injection of time into finite sets of instructions. Time and not space becomes the motor of change here.

To put it more simply: as digital computational models have become dynamic processes (driven by evolving and swarming algorithms), the Euclidean grid of discrete objects has turned into a continuous variation of form. What is at stake with these generative algorithms is that the notion of discreteness (parts and objects defined by finite sets of instruction) has changed, and now includes a model of interactive agents that evolve *in* and *through* time.

Architects Bernard Cache and Greg Lynn, for instance, have captured this nuance of algorithms or computational processing of form with the notions of the "objectile" and the "blob."[90] Objects, they contend, are no longer designed according to exact coordinates—points in space—but are complex curved surfaces, the slight variation of which cannot be controlled in advance. Hence an object is determined by an unlimited number of variations occurring through time, and is enfolded into an environment of differential relations, speeds, and intensities. From this standpoint, an object is always more than one and less than many. It is not the sum of interactive parts and does not remain a plurality of parts. As Lynn points out:

> With isomorphic polysurfaces, "meta-clay," "meta-ball," or "blob" models, the geometric objects are defined as monad-like primitives with internal forces of attraction and mass. A blob is defined with a center, a surface area, a mass relative to other objects, and a field of influence. The field of influence defines a relational zone within which the blob will fuse with, or be inflected by, other blobs.[91]

Lynn's computational architecture makes explicit that objects are not substances but acts and verbs. They can only be described in terms of vectors and temporalities. Blob objects have an aqueous spatiality, the unity of which is a folding of relations that constitutes a quasi-thing that is amorphous and uncharacterized. Far from being a substance that maintains its identity in space while moving from one position to another as if on a grid, objectiles and blobs are spatiotemporal events.[92] The unity of an object, therefore, is an ongoing process of infinitesimal happenings.

From this standpoint, blobs and objectiles show that there is no fixed space, because space is itself a topological surface of movement and variation. Rules evolve and change, are flexible and malleable. Space is then generated in time through evolving and interactive agents defined by vectorial forces. Space, therefore, is an envelope of differentials. As Lynn observes: "The prevalence of topological surfaces in even the simplest CAD software, along with the ability to tap the time-and-force modeling attributes of animation software . . . [supplants] the traditional tools of exactitude and stasis with tools of gradients, flexible envelopes, temporal flows and forces."[93]

One could thus argue that algorithmic architecture embraces the limit of computation, and reveals how the discrete logic of binary states undergoes a temporal inflection. This inflection, or deviation from a linear trajectory, reveals the qualitative transformation of parts as they become other than they were through interaction. And yet, if the becoming of an object only occurs through the qualitative transformation of its interactive parts, defined by the flexibility of algorithms and the temporality of processing, then how can one explain the indivisible unity of nonetheless discrete objects? One may wonder, that is, whether the continual sequences of algorithms can become a slice of spatiotemporality, an unrepeatable event that stands out from the seamless processing of generative algorithms and the information background of ubiquitous computing.

The next three subsections of this chapter will analyze theories of objects and of process in order to address these questions. In particular, these theories may help us to clarify whether an algorithmic object is to be conceived as a discrete unity or instead as the result of continual relations.

1.3.1 Unity and relation

The ontological problem that underpins conceptions of unity or relations returns, in algorithmic architecture, as a computational problem. Do organic forms correspond to mathematical sets? How can the continuity of vectors be split into binary "yes" and "no" states? Or can automated computation define objects in terms of relations? In sum, the question is whether discontinuous, disconnected, and atomic unity can explain the relational intricacies of more than one and less than many objects.

For instance, computational notions of objectiles and blobs are based on the temporality of processes and on the nonlinear (or differential) interaction of parts. It can be suggested that these notions do not describe the unity of objects, but precisely challenge the idea that an object is indivisible and determined by its spatiotemporal position on a grid. From this standpoint, an object always already fuses into another, or never remains the same. Here objects continuously undergo change. In fact, the question here is no longer how objects change, but how change defines what is to be an object.

Nevertheless, if change is to be our entry point into understanding algorithmic objects, then one could argue that change cannot explain away quantity, extension, order, and logic. In other words, algorithmic objects cannot avoid being determined by sets of instructions: an irreducible discreteness that does not easily match with the view that algorithms are the result of computational processes. In other words, the relational nature of each and any object of computation remains an ontological problem, which cannot overlook the discrete unity of algorithmic objects. In order to resolve this apparent paradox between unity and relation one may need to take a detour away from algorithmic architecture and explore recent theoretical debates about the ontological state of objects.

For instance, Graham Harman's object-oriented metaphysics precisely poses the question of how objects could be granted an autonomous existence that would not prevent them from entering true relationships; by extension, it is also a question of autonomy that the visions of metaphysics could support.[94] Whereas theories of modeling space and matter are characterized by an emphasis on self-organization,[95] object-oriented metaphysics explains that objects are actual entities that do not fuse into one another, and that do not continuously change.

In particular, Harman's metaphysical premises are that objects are autonomous and cannot be reduced to their components. They must therefore retain an indivisible individuality. Harman believes that even if a chair contains a complex number of elements and variations, it ultimately remains a chair. This means that the chair's qualities are irreducible to its construction, its material components, and its ideal uses. Yet he also claims that objects can and must become wrapped within other objects, and thereby connected to other universes. This form of relation, however, does not and cannot result in a total mimesis, where everything is linked to or can become everything else.[96]

Harman's object-oriented metaphysics moves straight against the current of the late 1990s triumph of system theories and rejects the imperative of connectionism by which all entities, at the micro and macro scale, are to some degree linked through others. Harman instead argues that "objects do not fully manifest to each other but communicate with one another through the levels that bring their qualities into communion. The level is not primarily a human phenomenon, or even an animal phenomenon, but a *relational* one."[97] In other words, only if objects are viewed as being indivisible and irreducible to other objects can their relations be explained.

Here relationality is defined not as a line between points, but as a multiplicity of levels that constitutes the fractal geometry of each and any object. Relations describe the "intermediate zone through which objects signal to one another, and transfer energies for the benefit or destruction of one another."[98] Levels are media. The latter allow objects to interfere with one another. Levels, therefore, are mediators not of objects in relation to others, but within objects themselves. In other words, relations are also objects: joints and glue, pipelines, tunnels, and crawl spaces, copper cable, fiber optic, smoke signals, quantum leaps, algorithmic processing.

From this standpoint, as much as relations are objects, so objects are not products of relations; they are themselves entities, possessed of internal qualities that also enjoy a relation of quantity. This is to say that objects are not only brought together by qualitative attributes that they all equally share. A quantitative relation may include, for instance, the algorithms that determine the chemical composition of a singular object, which can also be transferred to another without however reconstituting the same object. For example, water is a complete object, the qualities of which include the absence of color, smell, or taste and the natural states of liquid, solid, and gas. It is also constantly changing and in movement. However, the water molecule's chemical and physical properties are also shared by many other objects; for instance, it shares O (oxygen) with oxygen-based objects such as the atmosphere, and H with hydrogen-based objects such as stars.

Nevertheless, this is not to say that these aqueous qualities confuse the singularity of water with that of oxygen. Since qualities are intrinsic to these objects, they are indissoluble from them, and yet irreducible to this or that particular object. From this example, it is not only evident that qualities define the indissoluble singularity of an object; it is also clear that quantities and their relations define the singularity of an object. For instance, the molecule of water is composed of two atoms of hydrogen and one of oxygen. The hydrogen atoms are attached to one side of the oxygen atom, resulting in a water molecule that has a positive charge on the side of the hydrogen atoms and a negative charge on the other side, where the oxygen atom is. Since opposite electrical charges attract, water molecules will tend to attract each other, making water "sticky." But these numerical and electrical quantities also compose other objects, and relate objects together. Each and any object is indeed not only chemically composed of or related to another object; in addition, its algorithmic computations act to chemically constitute the operations of an object, which can also be generative of other objects and relations. This example could explain why Harman's theory of objects is relevant to the articulation of algorithmic objects, which are not simply quantities of physical qualities, but are themselves qualitative quantities.

Besides believing that an object is an indivisible substance, Harman, more interestingly perhaps, describes the object's multimediatic essence as being made of levels, or media. As already mentioned, these levels are objects. In particular, Harman claims that a medium is any space in which objects interact.[99] In other words, a medium is itself a space, and not simply a channel between spaces. Thus the levels of objects are themselves objects or media spaces. Hence, no object can be simply united by levels, as these latter are intrinsic to the objects themselves, like the joints of an arm, the brackets of a door, the glass of fiber optics, software protocols TCP/IP (Transmission Control Protocol/Internet Protocol), algorithms in computation, train tunnels, the cream or filling in a cake, water for fishes, oxygen for trees, body for mind, and so on. Harman insists that these levels-spaces-objects are worlds that cannot be ontologically determined by the synthesis of the senses, the bio-logic order of things, or the gravitation of the earth.

According to Harman, therefore, there can only be an indirect cause of relationality between the multimediatic objects. But this indirect cause is neither physical nor mental. It is neither an immediate sensorimotor response nor an explicit re-cognition. Instead, this cause is above all "carnal." It involves a sensual relation between objects or between the many levels–media objects themselves.

1.3.2 Qualities and quantities

As already mentioned, Harman's object-oriented metaphysics may contribute toward challenging the view that algorithms are merely functional or quantitative reductions of analog qualities and relational space. From this standpoint, an algorithmic object is at once a set of instructions and the relation between algorithms. Thus, algorithmic objects are always a discrete unity, and always correspond to a quantity. However, one may ask, what exactly is this spatial relation between algorithmic objects? For instance, is this space object a mere aggregation of algorithmic parts? According to Harman, only an empty shell can be defined as a mere aggregation of levels. He insists that the parts of an object space can instead be only vicariously active, implying an indirect, nonlinear, and asymmetric process of conjunction and disjunction of parts.

Yet the question is whether these distinct parts of an algorithmic object can be bound together, and yet at the same time be detached from its unity. In other words, if parts are not simply interactive components forming objects-spaces, then one has to explain how these parts can partake of the singularity of an object while autonomously entering other objects.

To address this question, Harman uses the example of an apple to describe how unity and disaggregation work.[100] An apple's sweetness, fragrance, color, and nutritional value are, to a large extent, distinct parts, autonomous attributes (which could correspond to the elements of an algorithm). Yet somehow all of these qualities are intrinsically unified in a single thing, even if they cannot directly interfere with one another (Harman states that the nutritional value of an apple has nothing to do with its color, for instance). The apple, as a complete entity, is not affected by its distinct qualities: it can lose its color and proteins and yet still remain an apple. It always retains its autonomy as this (actual) apple and not any other. In other words, this particular apple is not any other because its proper interior composition depends on the interaction of its qualities. However, Harman explains, this interaction is never *direct* and cannot give us *this* specific apple. If it were so, each element would become interchangeable with another, thus losing its medial nature and ultimately its *objectness*. It would no longer be an object but a link between objects. As Harman suggests, it is the task of indirect causes, which prevent the self-constitution of an ultimate subject, to glue together the macrocosm and microcosm.[101] This is why the relation of the object with its parts is not a transparent one. An object can connect with its own qualities and with other objects, but it can never be determined by its capacity of fusing with others into one.

Harman's insistence on the objectness of relationality suggests that relations do not occur in a void but are themselves objects.[102] From this standpoint, the relation between two objects does not simply entail their fusion, but rather the leveling of distinct objects. In other words, no single medium of interaction could exist between things; only many mediatic levels across scales, many media objects, could then define the workings of interactive computation. Against the seamless algorithms of ubiquitous computation that define the metamedium of all media as the (background) environment of already programmed interactions, Harman's object-oriented metaphysics seems to point toward a postcybernetic metamodel of complexity that exposes the irreducible elements/qualities of an entity as such, and its indissoluble, atomic, yet infinite incompleteness. However, these qualities are also elements of other worlds. The red of an apple can also define another apple or an altogether different object, such as a bag. These qualities therefore also expose an irreducible computational arrangement that cannot be contained in the simpler, universal axioms or the cellular automata described by digital metaphysics.

According to Harman, objects have intrinsic qualities that define their complexity and their incompleteness. From his point of view, algorithmic objects can therefore be defined in terms of qualities. However, as mentioned earlier, when speaking about algorithms or any form of data it is no longer possible to overlook the quantitative dimensions of these objects. In particular, algorithmic information theory has pointed out that these quantities are not finite, but rather include incomputable data or the extraspace of infinite algorithms. The space object or the relation between objects does not simply define an object in terms of quality, but must also include a quantity and an extension. In short, if an algorithmic object is not only a quality (temporal or perceptive) but also a quantity, then its relation to other algorithms must be defined in terms of quantities and not just qualities. The point here is to explain how indeterminacy or incompleteness in quantity cannot be exclusively explained in terms of qualities. It is quantity, and not quality, that has become the ambiguous protagonist of recent information theory, and this has served to point out that infinity is a matter of incomputability, which is now an immanent probability.

From this standpoint, one may need to reconsider Harman's position. Object-oriented metaphysics rejects the idea that an object is a unity composed of parts. Harman's philosophy does indeed challenge the credo that each and any object is mainly the results of interactive parts or components, either physical or sensory (an aggregate of particles or sense perceptions). Instead, an object deploys a fractional geometry of qualities, which are themselves objects and not parts, indissoluble qualities making the architecture of each and any object incomplete. In other words, objects cannot be reduced to the sum of their qualities: they are not constructed by human minds, scientific knowledge, or by patterns of perception-cognition.

This metaphysics of objects therefore also defies the metaphysics of universal computation, confuting the idea that simple rules or algorithmic automata can ultimately compute all physical, biological, chemical, and material objects via the evolution of parts that generate the qualitative complexity of forms. However (and this is the argument here) object-oriented metaphysics cannot help us to engage with uncountable information or quantities. If algorithmic objects do not simply constitute the universal language that can calculate everything, at the same time their qualitative elements cannot be isolated from sheer incomputable quantities. In order to defy the assumption that all qualities are reducible to simple rules, or that computation can explain the qualities of objects as the result of their interactions, it is important to question conceptions of quantification, calculation, computation, and algorithm first.

One may be right to observe with Harman that each and any object is irreducible to the secondary qualities attributed to it by the human mind, by perception-cognition, and by scientific knowledge; one could thus also conclude, in keeping with his philosophy, that objects have their own qualities, and that these latter are absolutely withdrawn from direct access. On the other hand, however, algorithmic objects are not only qualities. As information theory explains, they are rather finite and infinite quantities that are immediately experienced as data. To argue, then, for the discreteness of objects by explaining that the qualities of objects are themselves objects (i.e., qualities) may not be enough to grant actuality to algorithmic objects. What remains overlooked here is the actuality of algorithmic quantities that does not correspond to what can be finitely calculated: quantities that are not the same as qualities, but are instead more than realized qualities. To put it another way: the random and incomputable quantities of algorithmic objects cannot be replaced by the credo in infinite autonomous qualities. Incomplete qualities without indeterminate quantities instead will always already risk folding objects back into a complete axiomatics of denumerable parts or into the continual variations of physical matter. This is also to say that Harman overlooks the fact that parts are not simply finite elements and that they can be bigger than wholes.

As already discussed, digital metaphysics explains algorithmic objects as a set of probabilities or functions, made of exactly calculable components that lead to emergent complex behavior. But as I have also argued, this view must be opposed by the fact that there are algorithms that cannot be smaller than the objects programmed. Information never becomes fully transparent or reducible to a finite quantity, since parts of it remain incomputable. As opposed to digital metaphysics, which claims that cellular automata are the simplest and smallest programs by which the complexity of behaviors, patterns, and qualities can be generated, the metaphysics of incomputable algorithms instead grounds the reality of incomputable quantities in the sequential calculation of probabilities. From this standpoint, since incomputable quantities are indivisible, and remain, as Chaitin argues, discrete unities, computational continuity is conceived to be bugged by incompressible random quantities, where parts are bigger than the program devised to calculate them.

Harman's object-oriented metaphysics argues that there is a discrete reality for each and every object, the latter being irreducible to its essential qualities. This is also the case for algorithms, which define the finite states of any physical object as being made up of digital information. Similarly, this implies that the sensual qualities of digital images or sounds or spatial forms are not experienced as masses of sense data, but as objects. In other words, the qualities of a sensual object are not sensations, but are rather accidents that change while the *eidos* of the object remains the same.[103]

From this standpoint, the generative design of an interactive wall, as seen in the case of the Hyperbody's *InteractiveWall* and *Emotive Wall*, aims to correlate sense data to algorithmic objects by identifying digital objects with qualities. Such projects therefore seem able to reduce the incomputable reality of these objects to sensual qualities, which again are reduced to pixilated components of color, sound, and movement, thus confusing the transient qualities of perception with the *eidos* of the sensual object, i.e., the interactive wall. By contrast, object-oriented metaphysics explains that we experience only objects, and not masses of sense data:

> When I circle an object or when it rotates freely before me, I do not see a discrete series of closely related contents and then make an arbitrary decision that they all belong together as a set of closely linked specific profiles. Instead, what I experience is always one object undergoing accidental, transient changes that do not alter the thing itself.[104]

In other words, sensual objects cannot be reduced to parts or the atomic elements that explain their unity. Similarly, algorithmic architecture does not posit the reality of objects. "An object is real not by virtue of being tiny and fundamental, but by virtue of having an intrinsic reality that is not reducible to its subcomponents or exhausted by its functional effects on other things."[105] And yet, even if one could hypothesize that algorithmic objects could be understood as sensual objects, irreducible to their atomic components, the problem of how to define quantity remains. To argue that the qualities of objects cannot be quantified does not simply mean that quantities do not exist, or are irrelevant to the definition of an object. On the contrary, what is argued here is that discrete yet complex quantities, such as Omega, clarify for us that between algorithms there is no seamless computational continuum.

To argue that (incomputable) quantities are intrinsic to algorithmic objects is not, however, to sustain Harman's metaphysical schema, wherein the problem of quantity has been turned into a question of substance. In particular, Harman claims that "the reason we call these objects 'substances' is not because they are ultimate or indestructible, but simply because none of them can be identified with any (or even all) of their relations with other entities." Yet he cannot do without the notion of substance. "Every object is both a substance and a complex of relations. But if every object can also be considered as a set of relations between its parts or qualities, it is equally true that any relation must also count as a substance."[106]

In the age of the algorithm it is not, however, substance as grounded quantity but rather incomputable algorithms that have infected the computation of sensual and real objects. The liveliness and persistence of the substantial core of these objects is then split into a thousand fragments: into infinitely fractionalized algorithms that enter the operations of programming. Yet this fractal splitting of the united core of an object does not mean that there is no longer an object at all, and that only parts remain (i.e., algorithmic parts, by way of which the entirety of the object can be reaggregated). When sensual objects, spaces, walls are programmed, they are also exposed to the incomputable quantities that expose the unity of the object to deterritorialized signs. When a real object—a tree, for instance—is algorithmically programmed in a video game, its reality too is contaminated by the incompressibility, the nonchorality (lack of harmony) of discrete ciphers that are able to expose the algorithmic tree to its own infinite parts.

It is my contention that incomputable algorithms, and not information substances, are at the heart of programming cultures (from software design to molecular genetics, from nanodesign to social and urban engineering). Beneath the surface of ubiquitous computation—a surface composed of too much information, too many direct connections and interactions that are too smooth, all of which take place between programmed objects—there remains a contagious architecture of infinite parts, unsynthesizable quantities, and uncountable randomness that explodes within and between any finite kernels.[107] Since Harman's philosophy does not sufficiently help us to define algorithmic objects, one may need to draw on another notion of actual object, which instead accounts for the presence of indetermination in quantities. Similarly, it is also important to explain how algorithmic objects stand out from computational processing, or whether an emphasis on processes can account for the nuance of such objects. If computational processes involve recursive algorithms that ultimately generate complex forms, then one may need to explain how and why complexity already exists in the algorithmic object in the first place. To do so, the next section of this chapter will turn to Alfred North Whitehead's claim that objects cannot be explained by process, but rather exist as forms of process.

1.3.3 Form and process

All mathematical notions have reference to processes of intermingling. The very notion of number refers to the process from the individual units to the compound group. The final number belongs to no one of the units; it characterizes the way in which the group unity has been attained. . . . There is no such thing as a mere static number. There are only numbers playing their parts in various processes conceived in abstraction from the world-process.[108]

According to Whitehead, there are no numbers without group, no unity without process. However, as the quote above also serves to illustrate, Whitehead believed that there are processes of another kind that do not correspond to the world-process (the continual chain of actual worlds). Whitehead is specifically addressing the abstraction of mathematical processes here, but in this subsection I will discuss whether these abstract processes can explain the indeterminate quantities or incompleteness of algorithmic objects.

It may not be too easy to shift from Harman's object-oriented metaphysics to Whitehead's process philosophy: for against the idea that objects are born out of processes, Harman firmly maintains that the ontology of objects is irreducible. For instance, he questions Whitehead's process metaphysics on the grounds that it replaces objects with events. For Harman, Whitehead's work attends only to what happens to things, and not to what those things really are.

For Whitehead, the actual world is composed of actual occasions. These actualities are grouped in events, which become the nexus of actual entities that are "inter-related in some determined fashion in one extensive quantum."[109] Events therefore explain the togetherness of actualities, which Whitehead calls the "nexus." But every nexus is a component part of another nexus. The latter emerges as an unalterable entity from the concrescence of its component elements, and it stands as a fact, possessed of a date and a location.[110] Whitehead points out that the individual particularity of an actual entity, and of each nexus of entities, is also independent of its original percipient and thus "enjoys an objective immortality in the future beyond itself."[111] From this standpoint, Whitehead confutes the primary notions of space and time, and argues that only events, as nexuses of actual entities, are able to remain unrepeatable places with dates. In other words, actual entities are immanent events of time and space, and yet, as nexuses of entities, events go beyond *this* space and *this* time.

Nevertheless, Harman believes that this notion of event is determined by a transcendental cause, since events explain objects in terms of the effects of a process. He argues that process metaphysics always already defines the object (e.g., a tree) according to the effects that it has in the world, and not as a cause of itself. In Whitehead's metaphysics, he insists, the effects that objects have on others determine their existence. In contrast, Harman argues that objects and their relations are ontologically independent from their effects. For Harman the interaction of objects, therefore, can only occur indirectly or vicariously once the elements and qualities of objects become objects themselves. Thus, algorithmic objects should not correspond to the effects that they can engender in physical reality. Instead, interaction should expose how one object can become another, how one code can determine and change the milieu for which it was designed.

From the perspective of object-oriented metaphysics, the levels of interactivity implied in the *Emotive Wall* project, for example, are simply the effects generated by algorithms. Here, the interaction between objects cannot unleash new objects, but can only be reduced to the potential effects that algorithms can have on the wall, so that this latter interacts with the participants' movements. In other words, from this view the *Emotive Wall* is an example of a responsive architecture that is generated by the effects algorithms can have on the physical structure of the wall as it reacts to the sense data collected from the people involved. Ultimately, the *Emotive Wall* is the effect of algorithmic programming.

But how can algorithmic objects be more than an effect that one object can have on another? According to Harman, objects interact through their qualities. These latter are not simply projections that create effects, but are instead objects themselves.[112] In particular, Harman uses the notion of "allure" to explain that the relation between objects is vicarious, indirect, and hence unable to transpose effects onto things.[113] The concept of allure explains how qualities become things beyond any system of reference. Effects, however, can only derive from a system of reference, and convert objects back into mere secondary qualities.[114] Thus, in the case of the *Emotive Wall*, algorithms are not conceived as objects but as the projection of qualities onto the wall, which also remains the effect of sensorimotor interactions: the wall can bend, emit sounds, and create shades of color by responding to external stimuli. Hence neither algorithms nor the wall are objects here, but rather deploy a plethora of effects derived by a percipient.

Similarly, according to Harman, Whitehead's process metaphysics cannot explain how objects can truly interact, because qualities are merely attached to objects and are not objects themselves. These qualities, Harman argues, only constitute the events that explain the relation between objects. At this point, however, it may be necessary to get closer to Whitehead's view and highlight that events are not merely qualities deprived of objects; instead he conceives of events as a nexus of actual occasions defining an extensive set or a temporal series, an accumulation of actualities. Events, in other words, are "enduring objects."[115] Time plus space. Furthermore, events are both mental and physical, to the extent that thought and things have an equal but not equivalent metaphysical valence. For Whitehead, an event is the unrepeatable pattern that continues to happen and never remains the same.[116] For Harman, however, this temporality of the event is determined and linked too directly by the nexus of prehensions, which he argues is always already there to shape the object through projections,[117] and thus to reduce the object to shapeless matter without specific form.[118]

Harman's insistence on the actuality of things, on their indivisibility and ontological status as substances, may appear an alternative to the plethora of effects that define algorithms as mere projections on behalf of a responsive percipient. Nevertheless, his metaphysical schema may merely end up conflating, once again, the actuality of objects with their finite substance. Harman's position oddly resonates with a first-order cybernetic notion of objects, according to which these are determined by finite states. These objects will always remain what they are even though their qualities (and quantities) have nested with other objects and their infinite parts. If, as Harman tells us, the qualities are fragments, parts, parcels and bits that are themselves objects, then perhaps objects cannot remain the same after all. Nevertheless, it is true to say that he insists that an apple always remains an apple, even after its qualities (green, red, sweet, cooked or raw) have entered into relations with others and produced new objects (such as a green apple, for instance). This implies that the apple, were it to be computed, would be a finite set of algorithms, which despite interacting with others would always keep a traceable substance and original source. And yet the limit of computation suggests that objects have parts that are bigger than their totality; that there is always incompleteness within one set. This is to say that there can be no discrete and finite substance, no matter how much you break it down into autonomous qualities-objects. Not only is each and any object broken into parts that are autonomous from the object as a whole, but also these fragments can acquire degrees of completeness only after they have entered into relation with actual parts that have lost their constitutional origin. This is why objects are always partial things, and enter a nexus of events by which their irreducible parts can become new actual objects.

Contra Harman's critique, it is argued here that Whitehead's metaphysics of events does not determine objects by rendering them as the synthesis of the qualities that are projected onto them.[119] If, as Whitehead explains, each and any actual occasion is an assemblage of prehended data and prehending activities, then an assemblage is composed of parts-objects, which constitute an enduring object that acquires an epochal singularity. This singularity—which might be referred to as eventfulness—cannot be repeated, because the objects that define this singularity are partial, contextual, historical actualities. At the same time, however, if an event is a nexus of actual objects and not the result of projected qualities, it is because it corresponds to the eventuation of unprecedented qualities that go beyond the direct projection of the actual data.

However, is it possible here to understand relations to be both more than effects and less than the projections of a perceiving subject? How does Whitehead avoid equating relations with projections? In particular, one may wonder whether the notion of prehension can sustain the reality of objects without reaffirming the subjective (and phenomenological) experience of objects. To explore these questions is to probe how and to what extent Whitehead's process metaphysics can contribute toward defining algorithmic objects in terms of actuality and infinity.

According to Whitehead, prehensions are first of all mental and physical modalities of relations by which objects take up and respond to one another. As he puts it, "prehensions are concrete facts of relatedness."[120] He does not start with the substance of an object or with the perception one has of it, but confers autonomy on an actual entity's constitutive process of acquiring determination, completeness, and finitude from indeterminate conditions.[121]

Although for Whitehead prehensions are an external fact of relatedness, they are not mental projections but rather conceptual and physical relations:[122] not only concrete ideas, in other words, but also concrete facts. This means that the actual prehension of another actual object, or of its elements, changes the internal constitution (the mental and physical tendencies) of the prehended actuality.[123] From this standpoint, prehensions also account for how actual entities acquire determination or completeness from an indeterminate process of mental and physical contagion, or from the intrusions of elements from other actual entities. Whitehead calls this process a "concrescence of prehensions."[124]

Actual entities, therefore, are not substances or indissoluble objects. On the contrary, they can only *become* indivisible once the concrescence of prehensions affords an actual object that then becomes the subjective form of the data prehended. This process of prehensions is thus a process of determination, and what it determines is the actuality of data defined by the concrescence of prehensions. This is why an actual occasion is not eternal, but rather an event. It happens and then perishes. It acquires a subjective form of the prehended data and at once reaches objective immortality: it becomes an indissoluble event in the flow of time. From this standpoint, actual occasions are not effects of prehensions or mirrors of perceptions. On the contrary, they are led by their final cause to transform prehended data into a subjective form and into objective actuality.[125] The subjective form of the actual entity thus remains an objectified real potential that can be prehended anew by other actual entities. From this standpoint, the process of prehension is not a relative mechanism by which no object can as such be defined autonomously; instead, this process explains how actual entities become events, and thereby new spatiotemporal objects on the extensive continuum.[126]

As opposed to the relativism of ubiquitous computation, where everything is connected, Whitehead's process metaphysics is instead concerned with how indivisible or discrete unities can exist in the infinity of relations with other events, or with other actual occasions. This metaphysics does not offer us the option of simply merging or separating abstract and actual objects, but rather explains how infinity, indetermination, and abstraction are immanent to actualities. As Whitehead puts it: "The true philosophical question is, how can concrete fact exhibit entities abstract from itself and yet participated in by its own nature?"[127]

Each and any bit of an actual occasion strives for its own individuation by selecting or taking a decision about the infinite amount of data (the qualities and the quantities) inherited from past actual occasions, from contemporaneous entities, and from the pure potentials of eternal objects. Yet prehensions are always partial, since all actual objects at once select and exclude, evaluate and set in contrast all of the inherited data. In other words, prehensions do not at all coincide with a direct downloading of data on behalf of an entity, and do not constitute objects by projecting data onto them (or by way of what Harman would call the "house of mirrors," where objects become constituted by the reflections or the images of perceptive entities).[128] If, according to Whitehead, actual prehensions are the conditions of space and time[129] and are the markers of events, it is because the indissoluble atomic architecture of each and any actual occasion is imbued with indetermination. Whitehead's process metaphysics therefore suggests that events are a nexus of actual objects. These are unrepeatable events, and yet they remain incomplete because their objectified real potential can be prehended by any other actual entity and thus becomes other than it was.

From this standpoint, even if an object is what *it is* and cannot be another,[130] it remains an unsubstantial entity. An object cannot therefore remain unchanged from the material corrosions of its parts; it cannot stop bursting with its entropic chaos. Similarly, an actual object cannot remain an eternal form (the form of the apple) that physically reenacts itself and reproduces itself, as does an autopoietic system. Instead, an actual occasion maintains its objective determination, involving the prehension of both actual and abstract data. To put it otherwise, actual objects are not simply dissolved into a seamless process of projections, but are instead forms of processes: forms of an infinite number of infinities.

Whitehead in fact rejects the idea that processes involve the continual variation of a self-modulating whole. There could be no process without forms of processes, without conceptual and physical objects prehending the infinity of actual and abstract data. According to Whitehead, a form of process precisely responds to the question: "How does importance for the finite require importance for the infinite?"[131] A form of process therefore explains how "each fully realized fact has an infinity of relations in the historical world and in the realm of form."[132] In other words, a form of process defines how an object reaches its completion and becomes individualized, and how infinite potentialities, or eternal objects, enter actuality and determine eventful spatiotemporality. A form of process explains how unexpected worlds become added to already existing objects. Nevertheless, this form does not correspond to the sum of objects and the accumulation of qualities and quantities of data. The concrescence of the universe involves the concrescence of actual worlds that are imbued with eternal objects. Actual objects could not become complete, and there could be no event without the capacities of actual objects to fulfill the potential content of selected (or prehended) eternal objects, through which actual qualities and quantities can become other than what they were.[133]

Harman's object-oriented metaphysics contests Whitehead's need to make recourse to eternal objects or to potentialities in order to explain what actual objects are, how can they be related, and what defines a new object.[134] For Harman, the specific expression of an actual world is here and now, and cannot be explained away by eternal objects. Nevertheless, Whitehead's metaphysics does not simply replace empirical with transcendental causality, actualities with process, or facts with forms. Instead, it insists that there are immanent causes at work within an actual object: presentational immediacy and causal efficacy. While the former explains how prehensions are immediately taken by the present, causal efficacy refers to the reality of the past data that lurks in the background. If causal efficacy is "the sense of derivation from an immediate past, and of passage to an immediate future,"[135] presentational immediacy, the sense perception of things as they are presented here and now, is what is felt in the immediacy of prehension. Whitehead explains that the present locus is a datum for both modes of perception: it is an object of immediate perception according to the cause of presentational immediacy, and an object of indirect perception through causal efficacy. In other words, the double causality does not exclude the potential in favor of the actual, and yet does not simply merge the two causes together through material empiricism or transcendent idealism.

The two causes explain the immanence of infinity: eternal objects correspond to the infinite infinity of ideas, and actual objects deploy the infinite infinity of matter. It is when an actual entity selects certain ideas that a nexus becomes an event, and another actual object is added onto the extensive continuum. As Whitehead puts it, "a continuum is divisible; so far as the contemporary world is divided by actual entities, it is not a continuum but is atomic."[136] Eternal objects do not therefore glue actual entities together, merging all individualities into one continual process of projection. On the contrary, the extensive continuum as the general relational matrix of actual occasions is defined by "the process of the becoming of actuality into what in itself is merely potential."[137]

From this standpoint, if, as object-oriented metaphysics claims, not all objects are the same but every object is real,[138] then one cannot deny the reality of abstraction, or that nontangible objects have qualities, quantities, and relations selected by actual objects. The process of selection referred to is driven by prehensions, which can also be defined as modes of computing data from other actual entities and from eternal objects. Whitehead's metaphysics admits that actualities are infected with abstractions, with infinite parts that cannot be contained in an unchangeable substance. For this reason, Whitehead's conception of actualities may clarify for us how algorithmic objects can be actuals, and yet at the same time be imbued with incomputable quantities, with parts that are bigger than the whole.

Algorithmic objects could then be defined as finite actualities, not corresponding to an ultimate biophysical or ideal substance, but rather characterized by data prehended from the past (causal efficacy) and from the present (presentational immediacy). But this is not all: Whitehead's notion of the eternal object also explains how abstract infinities are immediately prehended by any actual form of process. His account of actuality therefore helps us to define algorithms as forms of process: as actual objects infected with the incomputable data of eternal objects.

Nevertheless, the relation between eternal objects and actual entities is not simply a matter of coevolution or structural coupling, as might for instance be claimed by an autopoietic approach to algorithms. Similarly, eternal objects do not generate actual occasions, but are "potentials for the process of becoming" of actual occasions.[139] Eternal objects are therefore immanent to and part and parcel of any actual entities, since the latter are precisely forms of process and spatiotemporal structures of data. Eternal objects are intrinsic to actualities, no matter how small and how inorganic the latter might be. Eternal objects are not the ideal continuity that links all actualities, but are indeed objects, despite being infinite. Whitehead's philosophy thus offers us an original view of infinity, which does not correspond to infinitesimal continuity between two objects but instead explains how eternal objects are infinite varieties of infinities nested *within* the infinite partialities of actual objects. It is this immanent ingression of eternal objects in the actual infinities of spatiotemporalities that deploys the workings of a contagious architecture, wherein actualities are hosts to infinite parts of infinities.

Eternal objects are real without being actual objects. The latter instead transform the pure potential of eternal objects into a real potential that is defined in time and space. Inasmuch as actual entities are causes of themselves, so too are eternal objects *causa sui*. This also means that their eternality is not grounded in substance, Spirit, or life. Similarly, infinity cannot be derived from finite actualities, because eternality is not flattened onto spatiotemporality. At the same time, however, eternal objects are not simply to be thought as universal qualities through which actualities relate. For Whitehead, eternal objects are ideas that are as real and as effective as any other physical thing. These ideas are at once discrete and infinite, since eternal objects are not equivalent to each other, but are instead defined by their own infinite process. Each eternal object or each idea is therefore not simply different from another. This is not simply a world of ideas: instead, each idea is constituted by infinite data that cannot be contained by a smaller entity or a totality. Eternal objects are incomputable quantities that cannot be compressed by actual quantifications (e.g., rational numbers). Eternal objects do not therefore simply guarantee continuity between actual occasions, because they are permanent objects that enter into actualities.

It may not be too ambitious here to understand Whitehead's notion of eternal objects in terms of random (i.e., incompressible and not arbitrary) quantities, as at once discrete and infinite. Eternal objects are anomalous entities that are not prescribed by a complete axiomatic system or explainable in terms of the indeterminacy of physical variations. While resembling the chaotic world of physics, these entities retain their mathematical abstraction. In other words, eternal objects are "objective and undetermined."[140] They explain how actual entities are never reducible to their actual parts, since a part is already a limit point of computation, the threshold at which discrete infinities advance into the actual world. As a form of process, an algorithmic object is therefore an actual occasion, the interior relations (or genetic inheritance) of which gives it an identity (a finished set of instructions) in the continual variations of qualities, while its exterior relations only attest to their capacities to become more than one. Eternal objects do not, however, reduce the concreteness of objects (i.e., their individuality and atomicity) to a transcendental being. Eternal objects like Omega are discrete varieties of infinities, inconsistent and innumerable entities, which never resemble the algorithmic sequences that they supersede. Indeed, these objects are bigger that any actual object, and while being ready to be selected by actualities, they cannot be reduced or synthesized by any particular actual entity. Instead, these objects are immanent to all actual entities to the extent that they irreversibly infect or virally program their atomicity.

From this standpoint, the interaction between algorithmic objects and between software programs, sensorimotor actuators, and physical prehensions does not simply constitute the architecture of the *Emotive Wall* as an entity that is deemed to change by way of changing levels of interaction. Far from being the result of projections from other objects' qualities, it is now possible to argue that the wall is not the end product of interactions between objects (algorithmic objects, actuators and objects of prehensions): on the contrary, each level of interaction is determined by a computational limit, or by indeterminate quantities. As Whitehead explains, eternal objects "involve in their own natures indecision" and "indetermination."[141] This means that pure potentialities, while being neutral, inefficacious, nongenerative and sterile, also remain passive as regards how they are selected and what is prehended. Whitehead clarifies: "An eternal object is always a potentiality for actual entities; but in itself, as conceptually felt, it is neutral as to the fact of its physical ingression in any particular actual entity of the temporal world."[142] This is why the grayness of the *Emotive Wall* is an eternal quality, which is not the same shade of gray that this particular actual response triggers on the wall. Its grayness in itself has no causal efficacy, no past and no future, no here and no there; it remains eternal and pure potential. It has no say as to whether it is prehended by a particular occasion of interaction.

As Chaitin reminds us, the algorithmic processing of data must include the infinities of incomputable discrete quantities between the sequential continuity of zeroes and ones.[143] Like incomputable algorithms, eternal objects are patternless and random, objective and underdetermined, and are ultimately incompressible into a simpler set of actual rules. Unlike the autopoietic ontology used to explain the generative self-organization of spatial form, where finite algorithms guarantee the production of complex forms by means of repetitive, periodic, simple rules, Omega defies the fundamental principles of mathematical calculation, implying that real numbers cannot be contained in smaller sets.[144] Eternal objects can be explained as real incomputable complexities. They are neither the cause of something nor caused by something else. Omegas are what they are: objects of pure potentialities, discrete yet infinite quantities ingressing (negatively or positively) into any binary computation. Like an eternal object, Omega is an incomplete cipher in the depths of each actual algorithmic pattern, but in itself it is neutral and underdetermined. Omega describes infinite varieties of quantities of color, volume, mood, depth, movement, concepts, and sensations. These are eternal qualities that can be selected by parts of actual occasions in their process of constitution, or concrescence. Eternal objects, therefore, are not simply expressed as effects in actual objects, but are the incomputable condition of finite, terminal, and unrepeatable actualities or nexus of actualities (events). Similarly, Omega complexities define the indeterminate conditions within which algorithmic objects are able to exist. Without the incomputable data of computation there could be no algorithmic architecture, since infinities mark the abstract spatium that affords binary design.[145]

Eternal objects are not only discrete infinities inside actualities. According to Whitehead, eternal objects also define how actual entities "enter into each other's constitutions"[146] and "express a manner of relatedness between other eternal objects."[147] Hence there are no vacuous actualities,[148] but only actual objects infected with the discrete infinity of eternal objects. If, as Harman complains, eternal objects are only there to guarantee infinitely regressive relations (making it impossible to determine the unity of any actual entity), it is not because relations are qualitative projections. On the contrary, eternal objects are there to explain why there is an infinite number of actual objects, and how these objects connect. Eternal objects define the ingression of incomputable quantities into actual objects and thus add new data to existing actualities.[149] Eternal objects are not there to guarantee a continual flow or smooth connection between actualities. Instead, these nonactual worlds explain how deep connections of ideas occur between the most varied objects and transform them.[150] This is why the relation between objects is not simply given by an ideal fusion, but rather implies the contagious architecture of actual entities (indivisible sets) imbued with eternal objects (infinite quantities): worlds belonging to irreducible yet immanent orders of reality, magnitude, and complexity. From this standpoint, algorithmic objects are forms of process, since they are actualities populated by infinite incomputable data.

The age of the algorithm, therefore, coincides not with the arrival of a new substance, but with the unleashing of data objects into programming culture. These objects not only calculate binary probabilities but have become speculative operators of incomputable quantities: data that cannot be compressed into smaller programs. Indeed, computation now occurs at the limits of calculability, probing into the realm of abstract objects or nondenumerable realities. The chaos of randomness is now the condition of calculation.

This is why computation can also be understood as a form of speculative reason, wherein algorithmic sequences are prehensions of pure potentialities (or eternal objects). But in order to discuss this notion of speculative reason, it will be important to explain quite what is meant here by algorithmic prehension. In the following section I will suggest that algorithmic prehensions ought to be considered in terms of aesthetics. It is impossible to speak about algorithmic aesthetics, however, without first questioning those notions of aesthetics that focus on computational beauty or on the elegance of codes, and also those that present digital aesthetics as being predicated upon a perceiver's framing of abstract data.[151]

1.4 Algorithmic aesthetics

Unlike computerization and digitization, the extraction of algorithmic processes is an act of high-level abstraction. . . . Algorithmic structures represent abstract patterns that are not necessarily associated with experience or perception. . . . In this sense algorithmic processes become a vehicle for exploration that extends beyond the limit of perception.[152]

In this passage, Kostas Terzidis observes that algorithmic processes extend beyond the limits of perception. Nevertheless, while it is acknowledged here that algorithmic structures do not correspond to what humans experience, or to perception, this section of the chapter will argue that algorithmic objects are actual entities and that they thus have physical and mental prehensions. Far from being qualitative impressions of the world or cognitive instructions that inform the world, algorithmic prehensions are physical and conceptual operators of abstract or incomputable data. Thus, on the one hand, algorithms are patterns of physical variables that stem from the circulation of the air, gravitational forces, the bearing of weight, volume, the geological nature of the ground, etc.; on the other hand, they are conceptual prehensions: operators of potentialities, not simply calculators of probabilities.

In particular, if algorithms mark the computational process of an architectural form that is as yet unrealized, then this is because these sequential sets are not merely preprogrammed symbols. If they were, one would be forced to accept that there is no novelty in algorithmic architecture, since there will be no conditions here for spatiotemporal becoming, or for new forms of process. Computation would remain a tautology.[153] Yet in contrast to such a view, we might quote Whitehead: "as soon as we abstract, so as to separate the notions of serial forms and of individual facts involved, we necessarily introduce the notion of potentiality."[154] This means that algorithms are the steps of a process of abstracting mathematical forms and individual facts, and that they are also conceptual prehensions of eternal objects, or potentialities, that determine the arrival of changing conditions in the process of calculation. According to Whitehead, conceptual prehensions are the feeling-thought of change before it actually happens. This implies no direct perception or advanced cognition of the future, but rather the selection of eternal objects by actual occasions: the becoming-form of irreducible qualities and quantities. At this point, one may wonder how these prehensive activities can help us to redefine computation as an aesthetic enterprise[155] that starts off in algorithmic programming.

It is difficult to discuss the notion of algorithmic prehension without rethinking what has been generally and historically understood as aesthetic computing.[156] In particular, aesthetic computing relies on the idea that the shortest program used to calculate infinite complexity is the most eloquent expression of harmony and elegance in mathematics.[157] To put it simply, aesthetics in computation, from this perspective, coincides not with notions of perception but with transcendental ideas of beauty, conceived as an ideal form and represented in geometric models of linearity and symmetry. This formal aesthetics is based on the predictability of results, where the more compressed the data, the greater the chance of patterns remaining regular, periodic, calculable, operational, and effective. Here aesthetics is not a process of prehension, but merely corresponds to already made and endlessly repetitive patterns, the simple functions that engender complex behaviors.

For instance, the algorithmic architecture of mathematician John Conway's *Game of Life* (1970) was composed by a two-state or two-dimensional cellular automaton, whose state was determined by the state of its neighbors.[158] Here all cells were indirectly and directly related to each other, and the process of computation was expressed by color change in each cell. The automatic deployment of the game was set by the initial setting of codes and rules, which gave rise to complex patterns that did not correspond to their original state. As yet another instance of the Turing machine, whereby a computational universe maps out all possible algorithms and their results, Conway's game computed anything that could be algorithmically calculated. It demonstrated how elegant codes and algorithms could be the simple solution to the complexity of systems oscillating between order and chaos. The process through which the patterns change over time coincides here with a computational process in which codes are themselves rewritten over time. This rewriting confirms that the initial conditions of the latter process did not predetermine its final result.

The *Game of Life* is based on the idea of a universal self-replicator. Small changes in its operations, defined by constant modifications to the configurations of the code and the rules, impart changes to the whole system. This extended Turing model works not at the limits of computation but rather within the frame of complete axiomatics, as it explains all outputs by smaller inputs. Conway's *Game of Life* therefore remains locked within an aesthetic of generative complexity. It makes use of the fact that cellular automata can serve to show that initial conditions are no longer retraceable once algorithms are set up to grow; but this is a model of computational aesthetics that overlooks the limits of an organic model of evolution of complexity.

A different approach was presented in 1997 by Jürgen Schmidhuber, who used the notion of Super Omegas[159] to describe limit-computable algorithms—the shortest algorithms (or the minimum description length algorithm) subtending a computation—as the equivalent of minimal art.[160] His notion of low-complexity art drew on Chaitin and Kolgomorov's algorithmic information theory to argue that the aesthetic modality of algorithms was equivalent to the subjective observer's enjoyment of the shortest possible descriptions of data. In other words, the more randomness (complexity or incomputable data) that was compressed into a cipher—for instance Omega, a limit-computable algorithm—the more beauty (understood in terms of the simplest or most elegant formula of complexity) was expressed. Like digital philosophers, Schmidhuber also argued that if the universe was computable, then there had to be a computational aesthetics running beneath all physical phenomena: an underpinning, foundational, mathematical beauty that could be expressed by the shortest of codes accessible to an observer.[161] Here the computation of probabilities occurred through the recursion of simple patterns generating complex behaviors.

From this standpoint, computational aesthetics is the manifestation of an elegant compression of complex data, which coincides with the synthetic point of perception (or the subjective synthesis) of random information. In other words, this model of computational aesthetics is defined by an act of cognition, the compression of data through perception.[162] Yet if the shortest code corresponds to the point of observation of the universe's simplest laws, then computational aesthetics can only celebrate reason as being governed by the shortest program, the compression of data to the simplest form. For this interpretation of computational aesthetics, reason, logic, and calculation coincide with the compression of information as the subjective limit point of perception. The more compressible, predictable, and cognized an algorithmic form is, the more beautiful it is. Any new pattern is defined by the capacities of cognition to shrink complex information into increasingly compressed forms of data. Contrary to Chaitin's point that Super Omega radically challenges the "Theory of Everything,"[163] Schmidhuber takes self-delimiting algorithms as an opportunity for a cognitive grasp of complexity: a perceptual (or, in his own terms, "aesthetic") point of observation that attests to a computational theory of the universe.

However, the aesthetic significance of programming cultures points in exactly the opposite direction. Ubiquitous computing, for instance, aims at constructing a megaweb of data that is easily transferable between mediatic platforms. However, and despite all efforts to fuse distinct platforms into one smooth plane of compatibility, there is no ultimate, finite set of algorithms, no elegant formula, and no synthetic AI able to compress all data stored and produced in distinct databases. On the contrary, the algorithmic processing of data cannot but face the ingression of incomputable information at the edge of each cognitive act of perception, an ingression that serves to release the aesthetics of computation from its limited conceptualization in terms of a sequence of logical steps, opening it into a speculative function of reason.

In other words, insofar as low-complexity aesthetics focuses on a cognitive point of observation that is supposedly able to grasp nondenumerable complexity as short sets of algorithmic instructions, it admits no novelty into computation, as it merely associates the latter with inward-looking cognitive patterns of repetitive instructions. Aesthetics corresponds here not to the prehension of eternal forms, but to the capacities of perception and cognition to reduce abstract realities to comprehensible mathematical forms. Yet these realities cannot be cognized, since, as Chaitin also suggests, Omega and Super Omega do not fit into a complete formal system and do not lend themselves to being mere spontaneous intuitions of real or transcendental infinities.

The incompressible random infinity of Super Omega supersedes the binarism of zeroes and ones by way of an infinite sequence of increasingly random Omega that pushes computation into an extraspace of data. It is argued here that self-delimiting Super Omega infinities are random surplus values of code[164] that drive algorithmic rationality away from preordained functions, and toward a speculative aesthetics that is defined by the conceptual prehensions of indeterminate infinities. In short, if all programming cultures share an axiomatic space of short programs that runs beneath all spatiotemporal complexity, this is because that space is contagious, composed of randomly increasing quantities, and a locus wherein new axioms are ceaselessly added across platforms, categories, and domains. This means that the surplus value of codes is not a spontaneous accident that is added externally to programming algorithms, and with the unpredictability of algorithmic form. On the contrary, surplus values of codes reveal the noncognitive prehension of incomputable objects or discrete infinities.

Putting it crudely, the age of the algorithm reveals the limit of rationality and perception. Yet this limit does not simply coincide with an underlying mathematical ground that holds the secret beauty of the universe. Algorithmic architecture is not merely an abstract data structure. On the contrary, it is a digital form of design that shows that actual algorithmic objects are immanent abstract structures. These actual objects are physical and mental prehensions of data, physical and mental forms of computational process. But as algorithms are physical prehensions of data (sets of algorithms that come before and after), so too are they conceptual prehensions of their computational limit. The aesthetic significance of algorithmic architecture, therefore, corresponds neither to the elegance of coding nor to cognitive access to mathematical truths. On the contrary, my contention here is that a proper engagement with the aesthetics of algorithms entails a notion of prehension. This questions the axiomatic and rational reduction of complexity to simple rules, as prehensions allow complexity to enter *into* existing sets of data. In consequence, algorithmic prehensions may clarify how and why computation has become a form of speculative reason.

1.5 Speculative reason

The commonly accepted model of computational aesthetics is based on the equivalence between the function of reason and the shortest of programs, cognitive observation and reductive axiomatics. Here aesthetics means the cognition of beauty and the mathematical understanding of data structures. But, as argued in the previous sections, algorithmic objects are more than finite axiomatics and less than perceptual forms. They are prehensive actuals that drive the computational processing of physical and conceptual data. As prehensive actuals, these objects suggest that computation does not simply imply the calculation of probabilities (as already programmed results) but the search for incomputable data, or eternal objects, that are selected and incorporated within them. This means that computation can be further understood in terms of speculative reason, which, according to Whitehead, challenges both the formal and empirical model of reason based on the mind and the brain. From this standpoint, computation reflects neither the working of a formal mind nor the changes of a biological brain. Instead, computation is taken here as an example of a speculative reason that is concerned not with using numbers to predict the future, but with following algorithmic prehensions to decide the present. Algorithmic architecture is therefore but one example of the way in which this computation builds the present through the prehension of infinite data. Algorithmic architecture is thus a case of speculative computing exposing reason, logic, and calculation to the power of the incomputable. But how exactly does the notion of speculative reason become relevant to computation?

Whitehead believes that it is an error to understand rationality as a result of the biological evolution of animal intelligence—that the biological evolution of the brain or the biophysical apparatus of cognition can explain reason. In short, he discards *tout court* the necessity of a bias toward the empirical for ascertaining the workings of reason. Similarly, he insists that rationality does not coincide with formal (axiomatic or ideal) operations of intelligence. In short, the function of reason is not to be found in the formal or theoretical systems that determine whether things are true or not prior to their actual occurrence.[165]

In particular, Whitehead warns us against the dominance of two main views as to what the function of reason really is. In the first of these, reason is seen as the operation of theoretical realization, whereby the universe is a mere exemplification of a theoretical system. The model of computation that views the latter as the ontological processing of complex data through the simplest of programs coincides with this view. Similarly, Whitehead rejects the metacomputational universe advocated by Leibniz (e.g., the principle of sufficient reason), as it specifically seeks to capture in the shortest functions—or finite equations—the infinity of worlds. Here the principle of sufficient reason reduces the nexus of actual occasions to conceptual differences, since this principle defines how differences can be represented or mediated in a concept.[166] According to Whitehead, this one-to-one relation between mental cogitations and actual entities is insufficient to explain the speculative power of reason, which is instead an adventure of ideas that is irreproachable by any complete formalism. Secondly, his notion of speculative reason is also divorced from practical and pragmatic reason, the view that reason is a mere fact or factor of the world, or is explainable as an immediate method of action.[167] In algorithmic architecture, this notion of pragmatic reason would constitute the critical view according to which computational modeling must account for the contingent dynamics of physical worlds. This may be the view that sustains interactive models of architecture, such as the Hyperbody group's projects *InteractiveWall* and *Emotive Wall*, where algorithms are correlated to physical data, thereby suggesting that software programs are only one of the factors in the architecture of a responsive wall.

Whitehead's study of the function of reason sits comfortably neither with the formal nor the practical notion of reason and suggests instead that reason must be rearticulated according to the activity of *final* causation, and not merely by the law of the efficient cause.[168] The final cause of reason explains how conceptual prehensions are not reflections on material causes, but instead add new ideas to the mere inheritance of past facts. Conceptual prehensions are modes of valuation that open the fact of the past to the pressure of the future. Final cause, therefore, does not simply replace efficient cause or pragmatic reason, but rather defines a speculative tendency intrinsic to reason. Far from deploying the effective power of reason, this speculative tendency, according to Whitehead, explains how decisions and the selection of past data become the point at which novelty is added to the situation of the present. In other words, reason is the speculative calculation that defines the purpose of a theory and a practice: to make here and now different from the time and the space that were there before.

From this standpoint, one cannot explain the universe solely in terms of efficient causation or by physical interconnections, as these dangerously omit any prehensive counteragency for which there can be no direct observation, intuition, or immediate experience.[169] For instance, the view of a physical universe determined by physical laws cannot account for the counteragency of conceptual prehensions to which "[the physical universe] owes its possibility of existence as a wasting finite organism."[170] These counteragencies are operations of reason directed by purpose and explained by final causation. This is why, according to Whitehead, the function of reason is "to constitute, emphasize and criticize the final causes and strength of aims directed towards them."[171] This means that the function of reason serves to unlock new possibilities within the order of things. On the other hand, however, reason also explains "the existence of a universe in dissipation within a finite time,"[172] and thus serves to acknowledge that things perish and that the universe as we know it will ultimately wither away.

It would, however, be misleading to equate this notion of final cause or purpose with a teleological explanation of the universe, since for Whitehead the function of reason is "progressive and never final."[173] This means that the purpose of reason is attached to the physicality of things but does not stem from them. Similarly, purpose in reason does not have to be exclusively attributed to higher forms of intelligence. For Whitehead, all entities, lower and higher, have purpose. The essence of reason in the lower forms entails a *judgment* upon flashes of novelty that is defined by conceptual appetition (a conceptual lure toward, a tendency of thought upward) and not by action (reflexes or sensorimotor responses). However, according to Whitehead, stabilized life has no proper room for reason or counteragency since it simply engages in patterns of repetition. Reason, as the interweaving of efficient and final cause, is instead conceived here as an *organ of emphasis* upon novelty.[174] In particular, reason provides the judgment by which novelty passes into realization, into fact.[175]

Whitehead claims that reason is speculative: an urge for disinterested curiosity, where reason only serves itself, rather than being a reason for (and of) something else. Speculative reason "is its own dominant interest, and is not deflected by motives derived from other dominant interests which it may be promoting."[176] A tension can be noticed here between a notion of reason as governed by the purposes of some external dominant interests and those operations of reason that are governed by the immediate satisfaction (prehensive self-enjoyment) that arises from themselves.[177] It would be a mistake, therefore, to associate speculative reason with functional adaptations—and evolutionary optimization—of the biological brain triggered by the environment, or by the natural selection of the best-adapted form. It would also be a mistake to conceive of reason as the result of interactive factors, whereby, for instance, the physical environment creates the conditions for reason to become dynamic, or for programming to become open.

Whitehead insists that while the history of practical reason is related to the evolution of animal life, speculative reason only belongs to the history of civilization.[178] In other words, reason is more than an organ of evolution, and does not serve the evolution of biological life. Contrary to recent claims, according to which artificial networks based on the neurological structure of the brain reveal the workings of perception and cognition,[179] Whitehead insists that the function of reason is deployed by actualities, and by a multiplicity of prehensions (of whichever kind and dimension). The function of reason is therefore a speculative affair: it implies a leap toward general reasons beyond that of higher forms of biological life, and beyond a specific method. Speculative reason defines a propensity for thinking that takes place at the limits of reason, and that enters the dangerous territory of prehending beyond the fact of the past.

Speculative reason is built not upon a simple observation, a single set of empirical data or actualization of a program. As Whitehead points out, "an abstract scheme conforming to the methodology of logic, failing to achieve contact with fact through a correlate practical methodology of experiment, may yet be of utmost importance."[180] Thus, Whitehead admits that the function of reason—as speculative reason—precedes direct observation or the point of cognitive synthesis of empirical data. As he clearly puts it, "Nobody would count whose mind was vacant of the idea of number."[181] The speculative reason for numbering numbers cannot be reduced to the practical or actual counting of what is observable. Whitehead continues: "The novel observation which comes by chance is a rare accident, and is usually wasted. For if there be no scheme to fit it into, its significance is lost."[182] And yet speculative reason does not subsume the fact of numbered numbers to mere ideals. In other words, speculative reason is not a function of static ideas, but a quasi-formal computation able to prehend novel data.

But while physical prehensions are explained by efficient causation, mental prehensions are their inverse pole. In particular, mental experience "is the experience of forms of definiteness in respect to their disconnection from any particular physical experience, but with the abstract evaluation of what they can contribute to such experience."[183] And yet mental prehensions are not functions of consciousness or cognitive actions. Whitehead explains that at the lowest stage a mental prehension defines the lure toward a *form* of experience, "an urge towards a form for realization."[184]

What is realized is the infinity of data that is already there in actuality. Higher forms of intellectual experience, however, can only arise from the double integration of mental and physical experience. This allows reason to become more than reason, and to enter a second order of mentality: "the appetition of appetition."[185] This second order does not conform to facts but exposes the immanence of infinity, adding novelty to the continuity of things in which reason is "degraded to being merely one of the actors in the efficient causation."[186] Speculative reason is instead a second-order mentality defined not by reflection but by immanent thought in reason, which "canalizes its own operations by its own judgments" and thus becomes the counteragent of repetitive experience.[187]

Whitehead argues that the aim of speculative reason is the production of an abstract scheme.[188] Yet speculative reason must at once transcend and utilize these schemes. For reason to be truly speculative, the schemes that are produced and realized must be able to encounter their finitude and limits. "Abstract speculation has been the salvation of the world—speculations which made systems and then transcended them, speculations which ventured to the furthest limits of abstraction."[189] But how does this notion of speculative reason contribute toward challenging algorithmic aesthetics and the rational logic of computation?

It is suggested here that computation must be reconceived from the standpoint of speculative reason: the production of abstract schemes. But in order to do so, computation must be made to confront its limit in the fact that incomputable algorithms add infinite data at the core of its closed formal scheme. This means that just as computation has to be rethought in terms of speculative reason, so too must computation be conceived in the aesthetic terms of algorithmic prehensions: the counteragents of efficient cause adding novel data to what already exists. This is a speculative and not an ideal or material conception of computation that breaks from the continual feedback of the "chicken and egg circle" of ideal deduction and empirical proof. Algorithmic architectures are not simply ideal structures that have to acquire physical boundaries, but are actual forms of processes that have an existence beyond any predetermined mental form and physical fact. Algorithmic architectures therefore are abstract schemes that involve the automatic selection, inclusion, and exclusion of infinite amounts of data, a form of process that constructs computational spatiotemporalities.

Whitehead warns us against the trap of pure idealism, and insists that reason could not become speculative if experience, fact, evidence, and possibilities were completely dismissed. Yet, as he clarifies, experience is not simply the result of self-reflexivity or cognition of data. Experience is always more than consciousness and less than perception. Unlike Hume, Whitehead argues that experience is not the locus of clarity, and similarly that the mind is not the locus of connection. On the contrary, clarity and attention can only be transitory: a few glimpses of clarity here and a few moments of attention there are all that can be experienced. Clarity, or the presentation of objects, cannot be separated from the stretched edges hovering around the here and now. Clarity is nothing without vagueness, just as light cannot exist without darkness. Indeed, if objects become clear and distinct it is only because they are imbued with an infinite variety of infinities, which contributes to the individuation or the character of actual entities. This means that experience does not coincide with the (perceptual or cognitive) synthesis of data, or the shortest measure of complexity. On the contrary, experience cannot but be immanent to the limits of what can be sensed and cognized, as expressed by the function of a speculative reason that injects incomputable rules into each level of programmed response and inference. For experience to happen, in other words, there must be an immanent prehension of incomputable data.

For Whitehead, experience implies the equal intersection of mental and physical prehensions, but novelty in experience requires the selection of infinity. This also means that speculative reason does not operate through intuition or direct access to eternal objects. Eternal objects, therefore, are not the qualitative attributes of actual occasions, as Harman claims, but are instead data objects themselves that cannot be made dependent on the locus of experience.

On the other hand, experience as fact, evidence, and specificity exercises authority over the conceptual prehensions of eternal objects. Whitehead points out that even the utmost flight of speculative reason must be equipped with a measure of truth, which works not to restrict or delimit potentialities but rather becomes a quasi-empirical condition for potentialities to add novelty to the order of things. Ultimately, for Whitehead, speculative reason without the wide world of experience will always remain unproductive of novelty. Yet the relation between ideas and experience does not simply involve the interplay between the abstract and the concrete, the ideal and the material. Ideas are as real as facts, and yet facts are infected with the abstract though no less real schema of eternal objects that are ready to inject novel data into experience.

Whitehead explains that facts are not simply there to become evidence of speculative thoughts. On the contrary, the authority of facts lies in their elucidatory power. Indeed, speculative reason is also a mode of scanning worlds to find evidence for this elucidatory power. Thus, reason moves beyond immediate fact, and ultimately aims to prepare experience for the ingression of irreversible data (incomputable objects) into actual occasions.

The epochal challenge of programming cultures is to venture into the infinity of incomputable probabilities (infinite discrete unities that are bigger than the totality of the whole sequence of algorithmic instructions) that lies beyond both the digital ground and interactive empiricism. The age of the algorithm precisely marks the moment at which the limit of data programming (the limit of computation) unleashes the incompressible nature of information into experience. Programming cultures are therefore instances of the unintended consequences of ubiquitous computing, the algorithmic background of which has been infected by incomputable probabilities. Programming cultures are the new operators of the speculative functions of reason, which point at a new aesthetic computation driven not by ideal forms but rather by algorithmic prehensions of random data.

It would, however, be wrong to view this state of incomputable chaos with naive enthusiasm. Instead, it is important to address the reality of algorithmic objects without overlooking the fact that the computation of infinity is at the core of logic, rationality, order, and control. This concern with incomputable probabilities, or with Omega, is therefore a concern with the transformation of automated functions of reason, cognition, and perception. This is not to be confused with a call for an underpinning mathematical ontology, able to adequately describe the truth of being. Instead, Omega shakes the mathematical ground of truth by revealing that the probability for infinity is an algorithmic affair that defines a nonhuman automated thought. My argument in this chapter has been that algorithmic objects are precisely these forms of automated thought, and that they unleash the immanence of a variety of infinities in computation.

One should thus partially reject Friedrich Kittler's suggestion that the end of the certainties of binary mathematics also marks the end of ontological thought.[190] According to Kittler, the ontological thought of technical machines needs to be seen through the trinity of commands, addresses, and data (processing, transmission, and storage). However, he also contends that the alliance of ontological thought and mathematics is now hiding a difficult truth, a truth that was already announced by parallel and quantum states of computation, which will soon replace big and serial silicon connections. According to Kittler, this technical transformation announces the point at which philosophy, as the ontological problem of thought, will reach a veritable end. Thus the end of digital media (the binary system of computation) also announces the end of philosophical thought, which is replaced by the triumph of practical reason, wherein thought is engendered by material, contingent, and varying processes (defined by quantum and/or analog computing).

It is easy to agree with Kittler that the end of the silicon-based computational model also marks the terminal phases of a metaphysics based on the binary model of thought (the formal axiomatics reducing physical contingencies to strings of finite algorithms). The terminal phases of this world may perhaps also imply the end of philosophy and its dichotomies between mind and matter, the ideal and the material. Nevertheless, it is equally easy to challenge Kittler's critique of software, insofar as it is based on the equation of thought to binary computation, and of philosophy to all modes of thought.[191] In particular, the relation in which thought stands to finite axiomatics as axiomatics stands to thought completely overlooks (1) that thought is not the same as binary computation, although automated thoughts are real; (2) that computation if anything is incomplete, and a Turing machine cannot offer a finite solution to the complex infinities of thought; (3) that algorithms are the conceptual prehensions of incomputable data or eternal objects, which have no biophysical ground in human thought (and in the ontological question of philosophy).

Regardless of whether quantum bits will mark the end of digital computation as we know it, or whether analog and quantum computing will expose thought to the material indeterminacies of atomic particles/waves, it still remains problematic to associate thought with a binary logic of finite states, and to make hardware the ground of software. Not only does this argument risk locking the ontological premises of thought into a monolithic philosophical system: in addition, it also overlooks the significance of incompleteness in computation, in terms of the capacities of automated thought to take decisions beyond original programming. As will be discussed in chapter 3, it is the immanence of chaos in algorithmic thought (or soft thought) that has come to threaten the idea that the automatic operations of computation are merely equivalent to or can be used to explain the *neuroarchitecture* of reason, and vice versa that neural networks are the spine subtending the architecture of thought. While the ontological claims for a universal computational machine proposed by digital philosophers pose cellular automata as the ultimate building blocks of reason, the incomputable algorithms discovered by Gregory Chaitin make use of the way in which the complexity of real numbers defies the grounding of thought in finite axiomatics. It is precisely the arrival of infinity in computational modes of thought that reveals the significance of speculative reason in the postcybernetic logic of control.

In chapter 2, this postcybernetic logic, in which axioms can be infinitely added into the automatic programming of physical variables, will be explained through the example of parametric architecture, which is an instance of the computation of topological (continuous) relations between parts. In particular, parametricism is an architectural style that uses parametric software to model urban space and behavior by including contingent relations in software programming. This chapter argues that the nature of formalism in design has changed, and that algorithmic rules are now exposed to *intended* indeterminacies built into the software itself. Parametric architecture, it will be argued, moves beyond responsive or interactive environments, because it is not just based on temporal variations (or intensive quantities) but rather, and significantly, on a new, quantitative ordering of spatiotemporal regions. Parametric architecture arguably offers a novel conception of space, which is described by the continuity of topological surfaces for smooth control. Similarly, however, it also implies a new "extensification" (a new potentiality for division) of abstract quantities into the spatiotemporal regions of parametric urbanism. *Parametricism* therefore reveals the algorithmic operations of a speculative rationality that is other than human, and is defined by the algorithmic prehension of physical and abstract data.

Whitehead's notion of mereotopology[192] and his atomic conception of time, which we will look at in the following chapter, will contribute toward explaining how the control of spatiotemporal relations now includes relations among wholes, parts, and parts of parts. This implies that control operates not only to ensure intensive or topological continuity between entities, but also to program the becoming of continuity itself. In other words, the question of control is now as follows: how can that which relates to itself become? To put it crudely, postcybernetic control is now concerned with the programming of events: with the nexus of spatiotemporalities infected with abstract objects.

However, if speculative rationality is at the core of postcybernetic control, this is not because its operations are rooted in biodigital systems of embodied cognition, based on interactive and neural network models. The attempt to eliminate all instances of abstract objects from the understanding of thought, perception, and cognition only amounts to ubiquitous computing's need to eliminate the immanence of abstraction altogether. As we will see in chapter 3, the computational design of spatiotemporality has been used to understand the cognitive and perceptual architecture of the brain, as modeled, for instance, through neural networks. However, this paradigmatic shift toward neurological architectures of thought (from notions of embodied cognition to notions of the extended mind), which routes thought into material substrates (whether these are animate or inanimate), only risks equating the reality of algorithmic objects with the mathematical or biological grounds of thought. This equation only works to disqualify the significance of algorithmic objects and incomputable probabilities vis-à-vis an understanding of thought that goes beyond the ontological premises (qua being and mathematical axiomatics) of philosophical thought.

This equation also fails to consider that although one tendency of postcybernetic control is to create a neoergonomic architecture of affective computation, another more subtle implication precisely corresponds to the failure of empirical functionalism to address the ontology of algorithmic entities and of incomputable objects without patterns. The more thought is embedded in computational apparatuses of cognition and perception, the more algorithmic objects unleash the incomputable data that cannot be synthesized, summed up, or simply instantiated in smaller programs (or in one totalizing form of thought).

Similarly, models of power relying on the continual regeneration of form and the autopoietic reenaction of thought as environment are no longer sufficient to explain how control has become an operation of prehension/*pre-emption*, to borrow from Brian Massumi, with power prehending (anticipating) its own limits/potentialities (the control of control). The advance of anticipatory architectures of power instead coincides with the proliferation of programming cultures (from DNA, bacteria, or stem cell cultures, to cultures of sounds and images, to time-based cultures or cultures of space modeling) that prehend the incomputable abstractions that follow fact, but which are not engendered by it.

This means that our postcybernetic culture is dominated not by the suprasensory bombardment of too much information, but by the algorithmic prehensions of incomputable data. This new form of prehension announces an aesthetic battlefield between the incompatible worlds of neurons and silicon chips, nanobots and blood vessels, the microcircuitry of computerized media and bodily temperature, software modeling and controlled gestures, actuators and programmed behaviors, which together deploy not a transparent apparatus of communication but instead a fractal architecture of events (an incompatible infinite nexus of spatiotemporalities). This aesthetic battlefield coincides neither with the presence of inaccessible real objects (as Harman would have it) nor with their readiness to be directly sensed.

In the age of the algorithm, space and matter have not become indistinguishable because the rules of modeling are common to all actual objects that originate from a physical substrate. If this were so, the age of the algorithm would simply be another instance of idealized empiricism, where actualities can only ever be enacted from a biophysical ground that is without any abstraction. Algorithmic architecture instead offers us the opportunity to conceive data in terms of spatiotemporal objects, which reveal the abstract architectures of space and time. But these architectures do not aim at predicting the future: instead, they reveal that immanent programming is at work in the present. Algorithmic architecture, as an instance of postcybernetic control, deploys incomputable objects in the programming of spatiotemporalities. This new mode of control, which places patternless data at the core of computation, will be the topic of the next chapter.