Divine Name Verification: An Essay on Anti-Darwinism, Intelligent Design, and the Computational Nature of Reality

Noah Horwitz

Published by Punctum Books

Noah Horwitz.

For additional information about this book
https://muse.jhu.edu/book/76445

For content related to this chapter
https://muse.jhu.edu/related_content?type=book&id=2653010
Let us say something more about evolution before turning to the Omega point. Chaitin himself, of course, recently has attempted to apply his own Omega theory to evolution.195 Chaitin however is still working within a purely Darwinian framework. That is surprising, given that in *Meta Math!* Chaitin notes the theory of Margulis, for instance, and “problems with Darwinian gradualism.”196 Chaitin himself notes that changes in genetic coding can have large effects, but says that “trading useful subroutines (this is called horizontal or lateral DNA transfer)” works best for doing so.197 Chaitin thus is clearly aware of the problems with Darwinism. And yet, in his essay “Life as Evolving Software,” does not speak about exchanging subroutines, for example. Instead, Chaitin believes that the complexity of life is like mathematical complexity. For this reason, he thinks the Busy Beaver function can simulate evolution. A Busy Beaver function is, of course, one in which the function always attempts to compute the largest possible num-

ber, given a particular computational framework. When the framework shifts, the Busy Beaver begins again to search for the largest number. Because math includes endless numbers, the Busy Beaver function cannot stop. Chaitin is thus trying to show that the randomness and incompleteness he found in mathematics will lead to unending evolution.

However, it is not clear that life is constantly trying to solve the same problem over and over again. Biology also does not have infinite complexity. It is finite and has only finite complexity. For Chaitin, math has infinite complexity because it includes the Omega numbers, but life is always about finite coded sequences and finite bit strings. While one can permute numbers indefinitely, it is not clear that biology has this limitless quality. Also, biology does not realize all possibilities. There may be no final truth in mathematics; but with organisms, if one has incomputability, the organism would be stuck in a loop or never be able to pass on its genetic code. Biological evolution then would have to deal with finite state machines and finite complexity. One would need to show from that how life forms constantly change and are altered.

If one makes biology into mathematics, the game is too easily won here. This is why Chaitin would have been better off trying to simulate lateral gene transfer and symbiogenesis rather than Cantorianism. Chaitin is an expert in programming languages. He would then have possibly found a way to uncover the finite set of rules that evolution is working with. Instead, the genetic code is treated as though it were computing numbers as a Busy Beaver function does. But no empirical evidence really supports that. Life also may not endlessly evolve. There may be a finite number of possible forms that at some point will be circled through. Chaitin adopts the Busy Beaver function because he thinks that if we treat evolution finitely, we have to treat it as a closed system. However, the finite nature of the genetic code does not make it complete. That is because we need to see incompleteness as a function of being itself. In the case
of the genetic code, it means more DNA letters can always be added to it. Any genetic code is an incomplete set.

The genetic code is not a Busy Beaver trying always to find the biggest number. In fact, the human genetic code has less base pairs than smaller organisms. Evolution is not about producing larger things. Sometimes—on an island, for instance—things shrink. Chaitin sees the Busy Beaver function always finding the most efficient way to compute the biggest number, given the mathematical point of departure, but because one can have infinities of infinities, the function will never offer a final answer. The Busy Beaver program never halts, because ultimately it is trying to calculate infinity. The genetic code is not. Even if it keeps permuting letters, that does not mean one has a better entity. Again, humans have less genetic information and smaller bit strings than other organisms.

Chaitin also, in his model, relies heavily on random change. In this way, the deck is stacked, but not in a way that simulates life. The Busy Beaver is constantly trying to compute infinity, and, at the same time, it has its bits being flipped randomly. Not surprisingly, despite the bit flipping (simulated random mutation), Chaitin’s function never ceases and never goes astray. It is not clear why, if Chaitin, is aware of the work of Margulis, for instance, he restricts himself only to random algorithmic mutations. In a book entitled Proving Darwin I received from him some time after this manuscript had been written that elaborates on the essay we are here discussing, Chaitin does refer to the work of Sandin in a sentence and says viruses could be the source of algorithmic mutation, but that viruses can introduce large scale change does not mean it must be random in nature.198

Now, Chaitin uses the Busy Beaver function to overcome any limits. Because the busy beaver is predesigned to always mathematically look for a specified thing, one does

not have to try out every possible mutation and every possible bit configuration. In this way, as well, the goal of the system is built in. It is always to get to the biggest number. If one has calculated the biggest number, one can then double it or square it or add it to itself. Chaitin is thus designing a way for evolution not to have to go through every possibility and actualize it, but, at the same time, in self-delimiting evolution, Chaitin prefixes the goal and stacks the deck.

What would have been interesting is if Chaitin had tried to find a way to simulate life’s development without doing a Busy Beaver computation. Chaitin, in his book, states that the role of the Busy Beaver computation is to weed out mutations that do not work.\(^{199}\) The genetic code would indicate in itself where to stop, as it does when it grows a leaf, for instance. Chaitin then could have shown why the genetic code functions the way it does. The genetic code, as a program, after all, has to indicate how to stop growing a leaf. It is not something that is externally imposed. It has to be coded in the compiled 0/1 of the genes. Also, if one truly allows random mutation to take place, then one will try to flip many bits at once. It is not clear this happens in nature. The Busy Beaver thus constantly eliminates the very randomness Chaitin allows in his system. Because there is a defined goal, there is a selector. But if Chaitin wants a Darwinian view, natural selection must be a random selector. It’s not going to choose the best thing in the sense of the most efficient or the most capable of doing an activity. The Busy Beaver program here is always selecting a specific type of program at each stage. Chaitin himself admits, in his book version of this vision, that his use of the Busy Beaver as an oracle enables his system to include “‘divine inspiration’ that enables our mathematician organisms to evolve, to improve themselves, to become substantially smarter.”\(^{200}\) In other words, we have, again, intelligent design mixed in.

\(^{199}\) Chaitin, *Proving Darwin*, 46.

\(^{200}\) Chaitin, *Proving Darwin*, 89.
And, as Chaitin notes (again in his book), having an oracle function here is a way to enable the computations involved to compute something it would not otherwise by adding, in essence, intelligence.\(^{201}\) After all, oracles are something intelligent programmers add onto software in order to filter and prevent errors. To think that oracles like a Busy Beaver would arise on their own is a bridge too far.

Of course, here also for Chaitin there are no time limits. In nature’s finitude, there is always a time limit. At most, Chaitin here has shown that if one sets up a function with a preset way of doing things and preset goal and then adds variation, one will have a system that grows endlessly and does so better than pure chance. Chaitin’s model is therefore too stuck in Darwinism. It ends up proving, like other simulations, ironically, the value of intelligent design versus Darwinism. Perhaps, at some future point, Chaitin will take more seriously the notion of the genetic code as software exchanging subroutines, for instance, and will produce a new model for us. If he does so, my argument here is, of course, that he will be led to the conclusions we have been laying out here. What this means is that the path not taken by Chaitin is to look at the genome as a programming language rather than the history of life as a Busy Beaver.

All programming languages can be compiled into finite bit strings. One can, for instance, as Wolfram has shown, reproduce snowflakes using algorithms alone. Very few challenge him on this and yet challenge the idea that life itself is also programmed. They challenge it because life can change in its programming, seemingly, whereas a snowflake cannot. The genetic code and life are computing their future states from their present states, like anything else. We need to understand how the genome is programmed to understand life’s development. But the computation itself might need to involve the genome plus cellular machinery and the environment. It will certainly, if we want to know why any particular organism looks the way it does pheno-

\(^{201}\) Chaitin, Proving Darwin, 46.
typically. Simply saying all is contingent is to say we do not know, but also to make a claim about how the world works that is at odds with what how it works in all other cases. This means, to truly see how evolution is programmed, we need to see how, given the initial sequence, that sequence changes over time, just as a series of sequences. That means following it from the first RNA sequence or first cell to the human being. In doing so, we need to see, following Margulis and Sandin, how bit strings merge and segments of them are transferred.

With Margulis, one bit string might be incorporated into another as a full subset. With Sandin, isolated sections will be integrated. What this shows us is that any bit string already has to be seen as divided into subsets. We also have to think sexual reproduction, where a new organism is not only the merger of two halves of bit strings, but also the re-sequencing with a minimum number of errors of those bit strings in the process. This is a mixing of programs (NKS 386). Such mixing is needed because the code itself, without such interaction, will not try out all possible programs. If all of life is compressible into the first cell, then even the viruses and bacteria adding new coding are part of the overall system of life. In this way, what appears as random mutation, code insertion, etc., is all part of an overall computation unfolding.

These changes in the programming have an important effect on the details of what the program produces. It is not natural selection driving things, but life computing its initial compressed state and elaborating it into Gaia. What these processes show is that there is not going to be a fixed mutation rate. Rather, one might have to sequence the genomes of entire organisms, trace histories, and then, based on that, determine how one bit string came to be another (unless one is able to simulate the very sequence’s unfolding). That would involve only looking at genomes’ software, separated from any hardware or environment. But that would be okay, insofar as the issue is not knowing why creatures look a certain way, but only how the software has
taken on its permutations. It would be like taking the different versions of Windows OS and asking how one gets transformed into the other and if there is some rule or set of rules for it. That would not tell you about the hardware used to run it and how it changed. It would not say what the ‘desktop’ looks like, but it can, in and of itself, express something about the coding. We are thus searching through the great book of life by listing all its letters and their permutations. However, we may see that the genome, in this way, is only part of the story, since it does not tell us, without being sequenced with cellular machinery, etc., what life is. It would be the evolution of the genome alone, but that in itself can tell us many things. In doing so, one will then look for the rules and operations that enabled it to occur. And from that research, the inherent laws of evolution will begin to manifest. But it will only be by doing the new kind of science advocated by Wolfram, without doing the laborious work of looking through the archeological remains of life.

What evolution itself is computing will here become clearer and clearer. This is a monumental process. We cannot know—in advance or from looking at the history of life in terms of, for instance, fossils or the menagerie of forms—what the computational operations are. Like with any computation, one goes from one state of bits to another. To understand evolution, we need then to know how, in fact, such genomes look as bits and lay them out to understand the algorithm at work. To do so is also not to ask about the origin of life, the origin of the software involved. That means viruses themselves have to be understood as not only inserting code from other species into our genome or code from the virus themselves, but also remodulating and sequencing our DNA. Unless one is to take all these events and transformations as purely contingent, one will need to do new scientific work.

Now, this is not to say that scientists have not already done some of this work. However, they have done so only under the influence of Darwinism. They have not thereby
fully understood that they are engaging in an archeology of software and not simply tracing chance changes. Today, this work is almost wholly focused on comparing differences and chalking them up to random mutations. Ultimately, one will need to engage in brute program emulation and repeat until one finds the right algorithm empirically, as a Wolframian approach would suggest. It is only ultimately by such a simulation that one can show that random mutation, for instance, is only a marginal phenomenon in the development of life. One runs the emulation.

If we need to start with the first replicated RNA strand, then there will be a tremendous number of possible programs to try. In this first sequence, we may find more of the future sequenced than we might now like to think. For instance, Spetner discusses how, from Talmudic and Midrashic sources, the RaDaL (Rabbi David Luria) put forth an evolutionary theory to show that animals necessarily change and evolve, but that there are 365 basic species that were first created and out of which they grew (NBC 212). All the species we see today are permutations and elaborations on these earlier forms. Each is then a microevolutionary development of these 365 basic forms. What might be true is that we can find 365 such basic templates as subsets or irreducible bits strings for these future species types in the first sequence itself.

What is important is that we remember that the “symbol strings” involved in the development of life itself are also informed by a “grammar;” such a grammar can specify how these symbolic sequences are to be coded for length, for how they can be combined, for how they can undergo inversion and substitutions, etc.202 One has to see these transformations as being like the transformations bit strings undergo in Turing machines, where the grammar consists of a finite set of rules to specify the way in which these grammars work. These grammars are not random, even if they are immanent to the bit strings themselves. If

---

they were truly random, then the entirety of life would not be part and parcel of the same living language. It would mean that, when looking at a human versus a tree, we would see two codes as different as Chinese and English.

These grammars are important, because the elements of the code have no determinate and inherent meaning. If the grammars here were random, as Kauffman contends, then we should not see the same genes at work in very diverse creatures.203 Lateral gene transfer also would not be possible. A grammar is inherent in the living language, but it is not random; otherwise creatures would develop in haphazard ways. To speak here of grammar is like speaking of algorithms, insofar as they specify how one string can be transformed into another. Like a computer program, the genetic grammar makes clear that there have to be constant ground rules about what sentences can, for instance, function as genes and what cannot. Grammars do enable these Turing machines to take an input and translate it into an output. Rather than random grammars, what we see is the combination of pre-existing words into sentences and sentences of longer lengths, just as we do in our own spoken languages.

If all things are here made up of bits at their most basic level, then how do we specify individuals? And how are there types? All is made up of letters, but—just as one can take a word and switch one letter to change the sense yet retain the same message and effect—so one sees that two things are identical in one sense, despite this sort of difference. We also see that new individuals emerge on this basis. The signifier emerges on the basis of the letter and also despite it, given the letter's meaninglessness in itself. Two children of the same parents might only be different by 30 letters out of billions. They are still two individuals, if only due to those 30 letters. Two books that differ by 30 letters are probably going to be classified as being of the same type. Some things seem identical, like two hydrogen atoms.

203 Kauffman, Origins of Order, 388.
However, identity really only means substitution. Two things are identical and of the same type if one can substitute for the other functionally and meaningfully without noticeable loss. Even hydrogen atoms are individuals.

As Manuel de Landa likes to point out, every atom is born in a star and formed by nucleosynthesis, one at a time, and has its own history. De Landa wants to trace the history of each atom rather than seek its meaning, its status as a bit or element. Substitution is the key to knowing two hydrogen atoms are of the same type. Any two can work with oxygen to make water. In fact, two hydrogen atoms might appear the same, except for being isotopes. There is always difference and differentiation. This differentiation itself is a product of the letter. The letter is always already different from itself, as it is split between itself and its place of inscription, its inscription and its absence. This is why individuals are always collections and sets. However, the sequencing can, at the very least, indicate what many of the subroutines are, where they come from, what function they perform, etc.

Of course, Margulis and Sandin still speak as though everything is contingent. Only by actually looking at the computation involved in life’s development will one be able to agree with that. People believe life is different than other computational phenomena because the code is changed. However, code changing over time can be a computation in the same way that a planet moving from one state to another is a computation. Genetic codes are of course extremely complex, insofar as the entire sequence is made up subsets. One cannot here, of course, also forget that the environment can also alter the code. At the same time, Margulis has shown us how changes in hardware can be passed on—in particular, changes in the cell’s hardware. That means it will not be enough to see the code and its subsets at work, but one must also see the code as interacting with other codes—the nucleus with the mitochondria, for instance. For that reason, it may not be possible to examine life without compiling the cell with all its machinery.
If the universe itself can be the unfolding of a single program, it is not clear why life cannot be, as well. The only way to do this will be for biologists to become computer scientists and make use of the resources found in that field. For instance, one will begin having to see various genetic phenomena as subroutines, literally, and to understand their functioning by looking at how subroutines work in computer languages. Physics and philosophy are themselves forced to become digital in the wake of the computational revolution now unfolding. Life is then but one special case where hardware/software split, but all phenomena are themselves computational and thus need to be understood with reference to programming.

Let’s first look at what is called an oracle. An oracle, in computer programming, is a tool for deciding if something is correct or false. The oracle checks outputs and then tests them in relation to preset criteria. In other words, oracles are a form of artificial selection. Natural selection simply names the idea that some things reproduce and others do not. If one wants, then, one can see an oracle at work that makes that decision. Oracles test things to see if they have failed or not. One can say thus that Gaia as a whole, for instance, has an oracle function in it that determines what will pass and what will fail based on the output, the genetic sequence. It is as though each individual organism is asking the oracle if it is one to pass on and how many will it pass on. And the oracle answers. The oracle could be pre-programmed to change how it answers and determines the values over time.

The oracle here is not random, but rather has a preset type that cannot and will not pass the test. For instance, a sterile organism will fail the oracle. The oracle is, in theory, that which determines what genetic sequences are meaningful and which ones are not. The oracle itself we are speaking of here is part of Gaia itself. There might be an oracle at the level of the cell, for instance, but it would have a much more limited function, such as simply testing
things in order to know to instruct another mechanism to come into play.

Let’s also look at the subroutine. This notion is very aptly named, as it names a subset of some larger set that codes for specific procedure or set of instructions. At that same time, a subroutine, as a subset, can be extracted and could fit into another program, if needed. Subroutines thereby can simply function like a computer within a computer. That is why they both can serve as models for understanding genes, but also the integrated and inherited machinery of the cell. One simply needs to make the cell the overall set. Subroutines are themselves activated and can be called into action and instructed by other elements of the code. What this means is that we need to look at the genetic code as being an assemblage of subsets that have specific computational abilities. It may then be possible to categorize them as one does, for instance, different cellular automata programs.

Also, one needs, in the code, a way to instruct it when to end. Computer programming already instructs us as to how that is done. One has a section of code that does that by, for instance, indicating each function must turn off. We need to be more shocked than we are that a butterfly’s genome contains all the needed information to produce both a butterfly and a caterpillar, including the chrysalis that produces the former. A butterfly would therefore form a very good example of how to understand evolution in capsule, not simply in the sense of seeing how a code has new information added to it via lateral transfer and symbiogenesis, but in terms of how the code itself, even without such additions, produces transformations.

Part of the key here is seeing the code itself as treelike in structure, insofar as it is subsets within subsets. One still has one overall set, whether that be the cell or the finite living language of all. The set theory used here is always finite when it comes to life. One will have chains, but the chains come to end, even if they are hierarchically organized. And if a butterfly genome contains information for
producing many different states, what is amazing is that the code has a latent code in it. Each genome contains information not being used, whether to build up things or to enable future developments. When the right instruction comes, this code can be activated. That is why viruses are like software patches being sent out by the code. Random mutation here is not at work, insofar as it is the execution of code already present that leads to structural change.

The concept of convergence already led us to believe certain things are inevitable, but if we begin to see how all life is inherent in the first cell, then all of life is a single organism. This means we must not just see specific genetic sequences as having subsets within subsets, but all of life as being a subset within subsets. Even an individual human is then a subroutine. The Darwinian view held that such subroutines are formed with no problem, as if it was the same as putting together any 12 letters to form a word. Just as not every twelve-letter sequence makes sense in English, so not just any sequence will lead to a subroutine.

It is always a question of finite sets when it comes to the living code of life, as well as when it comes to computer software. That is, the transfinite intrudes only in terms of the radical openness of the set to take in new info from a virus, for instance, and extensionality, insofar as the specific set is whatever is contained in it. If all is subsets within subsets, then there is some sort of hierarchy involved here. It is also an ordered hierarchy. Humans are made of cells, but cells are not made up of humans. As we saw with the empty set, there is some simple relation, at bottom, that is iterated throughout to form the hierarchical complexity. That does not mean each level shows more complexity, but that each rank is built upon the previous and that each level is iterated from the previous one. It also does not mean that at each level we have all possible permutations of the previous level realized. Life does not consist, for instance, of all possible ways of combining cells together. That is because there is program delimiting what will arise.

Insofar as we talk about sets within sets, any possible set
must be taken as being a concrete individual. A subset can be just as complex as a larger set. A subroutine can itself be a universal computer, but it will still be included in a larger set. It is a matter of inclusion rather than complexity here. When we look at life as being a series of sets within sets, we are not looking at it differently than we need to look at existence as such. Nonetheless, this view has particular implications. One implication is for evolution itself. Evolution itself is probably more so an arrangement of pre-existing subsets within the original self-replicating sequence, for the most part, than truly about the creation of genetic information. First, we see that there are various levels and scales of being. Any particular focus on an aspect of being is thus a matter of resolution. By resolution, we mean scalability. It is important here, though, not to think all things are fractals such that we will see the same pattern repeated necessarily at each level. It is more like a map, where we see at one level the continents and at another level the layout of cities. At each level, we only have one and the same object, insofar as we have iterations of the one Name. To look at a particular human cell is already to lose the human body. We are not speaking here of looking at the human body as being the same height as a human cell. If one zooms in on a map, one may know one is looking at a map of a particular state. Like with a high-speed camera and its ability to focus, one sees more and more structure come into shape the more one resolves the image. But we can also zoom out. What looks at first to be an independent set with subsets, such as a cell, in zooming out shows itself to be one of trillions in a human body, etc. The zoom in can only ever take us to the bit itself. The zoom out is unbounded, only in the same way as the universe is itself finite, but unbounded.

Because the universe is incomplete, we will not be able to stand outside of it. That means when we want to say what the whole set is we need to look at what it is in compressed form and look again to the letter, just as we did when looking for what it is at its lowest scale. We see then that there is a mystical identity between the largest and the
smallest here, just as there is a mystical identity between the empty set and the transfinite. This is what holism means here. It means that all is connected by way of the Name of God, by a set of letters, but also that the finitude of the created world is such that what connects all things is also the smallest and largest expression of them. The God’s eye view here is to see things as bit strings and to see all as related to one bit string. When we look at the first bit string, though, it is, in itself, meaningless. It is just a string of letters, like YHVH. It has, at this point, almost zero information. Almost nothing can be learned of it before its iteration, before the Bit Bang. It is inscrutable in and of itself.

Think here of what is called the Holographic Universe theory. This theory proposes, if I understand it, that the universe is bounded like a sphere, and on that sphere are encoded two-dimensional bits. The information then, so encoded, is projected holographically as the reality we are and exist within. Here, we need to ask what God sees. God sees then, first and foremost, the information itself, the bits. If we look from the outside in we, first and foremost, see the grid of information that is encoding the universe. If one wants, one can focus also on the hologram that constitutes an image of the world. On the inside, we are blind to this coding, as we are to the hologram itself. We are in the image that is projected from the surface. One of the founders of this theory, Leonard Susskind, has detailed in a book his debate with Stephen Hawking over the nature of black holes and if information is lost when something disappears into such a black hole. What Susskind tries to show is that information is never completely lost and erased. Information may get incredibly scrambled up to the point where we have no idea how to re-constitute it, but even black holes do not eliminate information completely. In this way, in principle, one can always produce what hap-

pened and reproduce the things, since, for us, they are made up of information. The information of the thing that falls into the black hole is encoded on the surface of the black hole in bits.

It is then no wonder that Susskind was the founder of Holographic theory, as here we have the same idea, in principle. One of the important aspects of this theory is that it founds the three-dimensional world on the basis of two dimensions. This is also necessary for a digital philosophy of the discrete. The discrete itself is two-dimensional in essence. God is thus not the set of all sets in the sense of a container. God’s Name is only the set of all sets in the sense of being the compressed form of all (rather than including) everything as an intensional set containing all other intensional sets. To say there is no set of all sets is not to eliminate the Other, but to say the world is not whole. And it is through the letter that keeps subsisting as letter and by way of its relationality that this non-whole is marked and marked as open.

All we have of God in this world is God’s Name. This is how we address and encounter God—through his Name and the effects of His creation. That does not mean we can only speak about the act of addressing itself as an act without reference. It means what is addressed is precisely what the name requires as its impossible reference—the nameless. At the same time, in addressing God, we are addressing the letters that the world itself is made up of and that are computing it. This is the metaphysical point, in terms of one of its most important practical implications. However, given that everything is a set within sets, the only bound is the Name of God itself.

We need to also look at how things are computed at various selected contexts. Resolution thus involves evolution and involution. Involution is when a set appears cut off from everything else. We see the set in its relative independence, as a system unto itself. Here, we are taking it as a bit string operating on its own. Take our favored example of a flame. One can abstract from all else and just look at a
flame and the program it uses to compute its persistence. The flame is but one thing transforming into another. That can mean it has its bits flipped or that it switches them around. A new rule is created or activated. The caterpillar becomes the butterfly, if we capture things over enough time. At one level, we have one genome, one set of rules. At another, we have new rules coming into play. At the same time, if we look at two parents uniting and producing a child, then we have two genomes becoming one. And what of symbiogenesis, where two things are combined without losing their relative independence and thereby not simply mixing their programs together? Here, we have convolution of part and whole.

Let us name two other concepts here. Revolution, following Badiou, is when we force out of a transfinite set a name that was previously indiscernible in it up to that point. Solution is when we sequence things to see how what a set contains is not some random grouping but a particular set of values. Given that here there are different levels and scales, we should not expect Gaia to be computing the same exact rule or algorithm at every level of life’s development. At the level of chemicals, that predates Gaia as such, there is one type of computational programs and set of rules at stake. One letter here is stitched to another, one more letter is added, a series of letters is doubled, and in all cases one needs to find the rule involved. When we speak of the world of viruses and bacteria, another set of rules might be dominant. The cell itself causes a major shift on how life operates, but sexual production does as well. There are different stages that emerge out of each other, just as life itself emerged from matter. These are all elaborations on the first compressed code that we could not have deduced but see in its articulation.

We do an injustice to the power of life itself when we reduce everything to the one note song of ‘natural selection.’ Recall how ‘natural selection’ was reduced to a tautological explanation for life, but it was also applied to all possible behaviors and activities. It thus became a simple
teleological explanation, where the final goal of all things is just passing on their genes. If we ask why humans developed philosophy, do we truly believe we can explain it in terms of the selfish gene? Nothing could be seen in terms of this academic Darwinism as itself a product of the code’s articulation or a by-product of other functions. Everything had to be seen as oriented to one final and tautological goal. Thus, almost nothing was explained, because everything was drowned in the acid bath of Darwinian teleology. However, Darwinism, allegedly, was supposed to remove all teleology and rely on blind processes. The explanation of things by the same final cause in all circumstances makes the explanation itself perfectly vacuous. If everything is done to pass on genes, then nothing in particular is explained. The difference between such a vapid way of seeing phenomena versus the way we are putting forth—that argues that such phenomena can be explained via their relationships to the elaboration of a computational code—are clear.

I also want to clarify, again, the difference between an intelligent design theory like that of Dembski’s and what we are proposing beyond what was stated in relation to Wolfram during our critique of CSI. When Dembski turns to elaborate his own theological model and to lay out his own metaphysical view, he models it on the idea of the ‘word made flesh’ rather than on a permutation of letters. This is because Dembski articulates his positive worldview on the basis of Gnosticism and Christianity rather than, as we do, on the basis of Kabbalah and Neo-Pythagoreanism. Dembski argues that, “the word in Christ was made flesh,” meaning the “divine logos” that creates all things is actualized by God speaking this divine logos (ID 225). Dembski thinks that God’s act of speaking “imposes a self-limitation on the divine logos,” but we need to see God’s speech as the permutation of the letters themselves (ID 225). That is, the genetic code, in its transformations, is how God speaks and, in speaking, creates. There is not an already existent and articulated divine logos but that the logos is itself built up
out of the primordial bit. Dembski thus sees God speaking specific, divine words that are like pure possibilities subsequently incarnated. The model of already formed possibilities existing in some Platonic realm that are then made flesh is the wrong model. As Bergson taught, such possibilities are a retroactive psychological projection, where we externalize what we can ourselves permute on the basis of the actual and make that ontologically prior. It is backwards and upside down to see possibility as primary.

It is not a matter of failing to work through all the pre-existing possibilities already there, but rather of creation involving a specific program, YHVH, that then elaborates itself. Dembski believes God’s divine speech is like a speaker of the English language who never speaks all the words of English, but we need to see English itself as a code built out of simple binaries (ID). Self-limitation cannot be thought without seeing it as involved with programming. It is not that God does not exhaust infinite possibilities, but that the transfinite itself is founded on the empty set. We thus can comprehend the divine logos, in large part because we will be able to emulate or simulate it. The world is not ultimate, because it is not complete and because it does not achieve every possibility.

Dembski also relies on a phenomenological model for looking at the world: “I look at a blade of grass, and it speaks to me. In the light of the sun, it tells me that it is green” (ID 232). Such a view looks at things only as parts and wholes. A blade of grass is a whole and a unity, a thing, which emerges and is grasped by consciousness. To engage with it truly as part of the logos is to find the program for that blade of grass, to understand it as an iteration of a simple set of rules over and over again—a set of bits, perhaps irreducibly complex, but still a set. By using a phenomenological model in this way to engage with the divinity of creation, Dembski treats information in a semiotic sense, as signs representing something for someone, us. However, information is, in itself, fundamentally meaningless. Shannon information is not semiotic in this sense, as it
is syntactic information, differentiability, and not a reference to a thing. Even semiotic signs are themselves dependent on such syntactic information.

Dembski is right that “creation is a gift,” but it is given via the letter (\textit{ID} 234). It is a token. This does not rule out that prophecy itself interprets the will of God and thereby deals with signs. As we saw with the Sinai event, the revelation there was an overwhelming voice that, in its infinite nature, dispensed to the prophet a sea of letters rather than pictures and images.