The Making of Code World

or

Time for a Language for Talking about a Language for Time.

$\{10^{10}\}([10^{10}]=[10^{11}])$.

My name is Mark White. I am working on a math related project, and I am looking for someone to help me. I do not know exactly what I am looking for, but this person must have good mathematical skills – specifically group theory and number theory - and perhaps some knowledge about computer language creation. Basically, I want to define a new system of counting that would lead to a new form of math. Nothing complicated. A background in biology is not important, but it might be helpful. This project is primarily "for fun," but it might also be seen as collaboration toward a paper to be submitted to mathematical, IT or life science journals. In the short term, however, I intend to use it to advance a work of fiction, so it should be loads of fun either way.

My background is as follows: I received a BS in Geology in 1985 and a Doctorate of Medicine in 1989, both from Indiana University. I have participated in the publication of a handful of unimportant scientific papers. I have no mathematical training other than what I have picked up as a hobby. I consider myself a competent amateur computer programmer who sold a little bit of his software many years ago. I am a serial inventor, and I have an eclectic mix of experiences in business. I have practiced emergency medicine for 25 years. In short, I am not a mathematician; I am not a scientist in any real sense of the word; and it remains to be seen what kind of writer I might become.

Like I said, the project I am working on involves a work of fiction. It is a fictional story about a real theory. The story is about a mathematician who develops a formal mathematical theory about aperiodic crystal growth one hundred years ago that is re-discovered in the twenty-first century, and then discovered to have been proven correct fifty years after its proposal - but totally ignored. The theory went unrecognized due to a massive conspiracy by beloved historical figures in science. It is historical revision plus absurdist humor in the spirit of Kurt Vonnegut or Charles Portis – my two favs.

The main premise of the fiction is that the theory could have been proposed one hundred years ago. I started writing it on a whim, and then the rest just wrote itself, so to speak. I have long dreamt of writing a novel, and now it appears that I will. Although I never fancied myself much of a writer, this theory of mine forced me many years ago to write several books. It was a vain attempt to describe my theory in a way that somebody else might understand it. So it has always been somewhat of an

obsession that I get people to understand it. They clearly don't want too, and I'm pretty sure that most people aren't able to understand what I am saying – no matter how I say it. I did prove, however, that I could hammer these books out relatively quickly. This book you have started reading is yet another example of one of those books, and I hammered it out fairly quickly too. This book reflects the contours of the thought process I went through to understand the main literary device I will employ in the novel.

I find this book necessary for three reasons: First, it allowed me to better understand the contours of my own theory in the context that I am now presenting it - mathematics. Second, it provided the main plotline of the actual novel. Third, it takes the load off when it actually comes to "explaining" the theory in the novel. The novel is a first person narrative, and the narrator is incapable of understanding the actual theory; he is only able to understand the contours of it and how it dictates the actions he is describing. I want to keep the techno jargon to a minimum; what little there will be will be merely reported to us by this layman.

As I was doing this, two things occurred to me. First, the theory itself could very plausibly have been proffered one hundred years ago. Second, it would most likely be seen as an entirely new branch of mathematics. It is highly likely that as a branch of mathematics it would have been seen to have little use. But if the theory had been predicted to be useful for exactly what I am using it today, it would not only be seen as useful, it would have been seen to have been "confirmed" many times over. It would surely be a part of our everyday language now. That's a pretty cool theory, if you ask me.

As it went, this theory of mine was put out there about ten years ago, and it just kind of laid there, in the parlance of our day. The general consensus seems to be that it has no worth. Naturally, I disagree with the consensus that it is worthless – it's worth something to me - and so I am now working on the story to try to make my point in an entertaining way. I want to tell the story in a way that a layman might enjoy reading it. The story is science fantasy, of course. In this case the fantasy is that my theory and the accepted theory that currently blocks it somehow had a chance to compete side-by-side, which they now obviously do not. That ship sailed a long time ago. That's why it is fantasy.

The theory itself can be stated in its most simple form perhaps this way:

The "genetic code" is a "language" being executed today by molecules. The language preceded the molecules.

I am forced to use a lot of scare quotes here because the words that we normally use to describe these common things logically requires that you know what they are supposed to mean, which begs the question of what they actually mean. It is common knowledge today that there is no language other than that represented by the molecules themselves. In other words the accepted theory that currently blocks my theory from consideration is its logical converse: The genetic code is a collection of

molecules that is not in any real sense a language. Even if it were a language, the molecules must necessarily have preceded it.

Another colloquial way to put it that will make sense to any 9th grader is this: The genetic code today is taken to be a spreadsheet of molecules. So which came first, the spreadsheet or the molecules? In other words, did the molecules give us the spreadsheet or did the spreadsheet give us the molecules?

A more abstract way to ask the question is this: Can there be – in any circumstance - information without language? If life is a system of molecular information, must there first be a molecular language to record and transmit molecular information? If so, what could it be? What, precisely, is molecular information? Since life has come to be seen as a system of molecular information, what is life?

So in general the theory is an existential one, but the many consequences to its acceptance or rejection are extremely pragmatic. That is why I think it deserves a fair hearing, yet it has never been given one, in my opinion. It has merely been taken as a scientific non sequitur.

So you can see that my little theory simply is that the spreadsheet came first. It HAD TO come first. The reason I say this is simple enough: I invented a better spreadsheet. Then I started to ask myself what it means to be a "better" spread sheet. The answer, of course, is that it means quite a lot. People can only see the spreadsheet now, but they cannot see the language that I first discovered that leads to this spreadsheet. Knowing the language first makes the selection of spreadsheets all important. In fact, I think the selection of spreadsheet has more meaning than the actual molecules filling it. I believe that there is in fact a best spreadsheet – my spreadsheet.

The basic problem that I have now is that very few people actually understand the theory, and the reason, I suppose, is that I have had a hard time formalizing the theory. I certainly haven't done a very good job of explaining it. One might view this little project as one of formalizing the spreadsheet. I am going to demonstrate some first principles that lead to a spreadsheet, and then show how these same principles are behind the spreadsheet we ought to seek. Then maybe a debate can actually take place. I'd surely love to finally hear that debate, lots of high drama and low comedy, no doubt.

Here is how I objectively knew that my spreadsheet was better:

- It is perfectly symmetrical.
- It is maximally compressed.
- It exists in a physical form that is isomorphic with all known forms of molecular information.
- It is one part of a two part system of a very simple language of shapes. The first part is isomorphic with the genetic code, and the second part is isomorphic with the "language" used by the double helix to replicate itself.
- Perhaps of lesser significance – it comes with a pretty cool toy.

Based on this, the overall goal of the fiction I am working on now is set out in the short prologue that I have already written:

There are many Life sciences, and they are quite diverse. For hundreds of years the Bible was the organizing principle behind the whole scope of life sciences.

In 1859, Charles Darwin did his part to change our perception of the role of the Bible in life science. He released his magnum opus, The Origin of Species, which lay out his description of and evidence for natural selection - the logical mechanism by which life arises and evolves on earth. It is conceptually simple, an abstraction born of pure mathematics. Many a scientist has considered God to be a mathematician, and a younger Darwin was a religious man. Regardless of Darwin's intent, natural selection is the organizing principle of the life sciences today.

In 1953, James Watson and Francis Crick described the double helix. This event was vaguely perceived as mathematical too. The double helix was an exotic name for an exotic shape that held the key to unraveling an exotic code. Arguably, this discovery has had a more profound impact on mankind than did the spelling out of natural selection. The double helix formed a gateway for the explosion of new ideas, techniques and technology in all of the life sciences. The double helix was born during the rise of digital computers. They are fraternal twins, and together they have reshaped our view of life on earth. Although natural selection remains the organizing principle, the double helix is undoubtedly now the leading icon. It is a touchstone for thought about life in our digital age. As such, it has clear advantages, but there are profound problems with using the double helix in this way. In fact, the double helix and all it represents is now doing us more harm than good.

In 2004, I built a device that consists of a tetrahedron (or more fully a cube) that rotates around or within a dodecahedron. I called it Code World. Then I discovered something unexpected and quite remarkable: Compared to the double helix, Code World is a better icon, and it is a far better touchstone for thought about life on earth. Life sciences across the board will benefit from replacement of the double helix by Code World.

Code World, in fact, will now serve as a better touchstone for life sciences than did the double helix. This has yet to become obvious to everyone, because things like this take time. But while we wait, it is helpful to know the full and fantastic story behind how I came to build Code World, and then how I came to appreciate it for what it really is. You see, I didn't exactly "invent" Code World. You might say it was given to me, or more accurately, I found it.

This is that story:

So that is the basic goal of the book. It sets the stage for our narrator to tell his utterly fantastic story of why he came to believe that Code World will become the icon of life sciences. Just think how many toys he would sell.

There are two premises embedded in the story. First, my theory could have been conceived and formally presented one hundred years ago. Second, had it been presented one hundred years ago, it would have easily been accepted at some point starting about fifty years ago, and it would have been empirically confirmed many times since then. Also, it would have had a huge impact on all of the life sciences and the way we think about them. It would have been embedded in the language we now use. It would have been the default icon. The double helix, when it was finally discovered, would merely have taken its logical place within it.

I firmly believe that today our language is dysfunctional in so many ways, primarily because we perceive two separate icons of life sciences – the double helix and the genetic code. My little toy – Code World – combines the two into one, and it turns the dysfunctional language inside out. It is more meaningful and useful than either of the two existing icons is alone. It is a literal combination of the double helix with the genetic code interacting in the language of life. How is that even possible? Why is that not significant? In what World could this possibly make sense?

As you read this, two questions will pop into your head: How can this guy do this, and why does he need any help doing it? The answers are simple. I was born a mathematician, and I never had any interest in math. I don't do math; I think mathematically – always. I don't solve a math problem; I see it. If I can't see it, I can't solve it. Despite a preternatural ability to solve seemingly complex math problems, I suffered the "show your work" curse in school. Show my work? You asked me a simple question and I gave you a simple answer. You want I should draw for you a picture of my brain? It's like seeing colors; you either see them or you don't.

I believe I was born a mathematician due to a combination of genetic defects. I see in mirrors, I have a horrible memory, and I have a fabulous imagination. This is the basic toolkit of a born mathematician. If you can't tell the difference between forward and backward, you learn tricks for identifying relationships between things, like the letters b and d. The proper relationship, which I discovered much too late, is best illustrated in the word 'handbook,' which is ironic, since the alphabet got it wrong. In other words, if you can't remember even simple things, like the alphabet, and you can't see the difference between one thing and another, you learn ways to figure it out on the fly.

Another example (which will piss off my dad if he ever reads this) can be seen in a simple card game. When I was a little kid we played a game over the holidays with family. We called it Concentration. You lay out all of the cards face down in a 7x7 grid, and then take turns turning two over. If they match numbers, you keep them; if not, you turn them back over. Ostensibly, it is a memory game, but it has less to do with memory than it does with math. If you see in math, the only thing you need to remember is which cards have been turned over. Some cards naturally appear to a normal person to be a "hit" when they are turned over. For instance, an ace in any corner is always easy to remember. But to a born mathematician, every card is always a hit. An ace or an eight, a seven or a king is always a simple hit in any corner, if you are used to counting backward and forward, if you are used to "remembering" things only on their relationships to other things. If you see in "shades of a hit" then every card becomes a hit in every position of the board. The board and pieces interact to do simple math with the game.

One more personal anecdote that I think best illustrates my point that I am a born mathematician who is lousy at real math and therefore will benefit from the help of a real mathematician. When I was an undergrad I answered a question in the college newspaper about a math problem: Which number is larger, 10^{100} or 100! If you do not immediately see the right answer, you are probably not a born mathematician, but it doesn't even take a born mathematician to see the right answer. It is the same as asking, which number is bigger, the little one or the big one. Most people will try somehow to "do the math." A trained mathematician will "solve the problem." There is work involved either way. The born mathematician doesn't bother to do any work because it requires none.

The trained mathematician solves it quickly by noting that $10^{100} = 100^{50}$, and 100! Is the product of (1 X 100) X (2 X 99) X … which represents 50 numbers, all of which will be bigger than 100. This is not a hard question; it is hardly even a question. Why is this even a question? All you need to be able to know the answer is to read the question. Read it this way: Which is bigger, one hundred tens or more than 100 tens? Clearly, there are more than 100 tens in 100! Why even bother showing it? We could easily show it this way $10^{100} < 9!^{10} \times 10^{90} < 100!$ Perhaps a computer programmer prefers to read it this way:

$10^{100} < 16^{104} < 100!$ After all, we can easily find 104 sixteens between 16 and 96. The reason I bring this up now is primarily because of the experience I had when I tried to give the answer to the math department. Apparently, they'd never thought of it quite this way. Seriously? It took a disturbingly long time for me to explain it, and then they put a considerable amount of work into showing me the answer I'd given them. They were amusingly enamored with the answer and the work involved. I don't think they ever really understood the answer. Maybe they did. Who knows.

The point of this particular anecdote is extremely subtle, but extremely important. Ostensibly, the question asks about the relative sizes of two numbers, when in reality it is about the relative power of two number lines. It now becomes purely a question about the power of two languages for counting. The first "number" is counted in our native tongue in its familiar base. There is hardly a difference between the line and the actual number. In other words, it asks us to "count" to 100 by powers of 10. Each step just adds a zero to the 1 – a novelty of our chosen counting language. We can quickly and easily write the actual number. It is simply a 1 followed by one hundred zeroes. The other line is unfamiliar and involves a counting language with which we are entirely unfamiliar. It then asks us to count to 100 with this line in this foreign language. What's a poor boy to do? This does not stop us from knowing that we are still just counting. The simple answer is that it is still exponential counting, merely with an obviously higher base. There surely is a natural base for all exponential counting on the factorial number line, and it surely is much higher than 10/100.

The job of a mathematician is not to solve problems, but to find general solutions to different kinds of problems. The first task is to recognize the kind of problem. This particular problem has an easy answer because it has an easy general solution. The "classic" solution has us reduce the powers until the bases obviously cross. They cross after the first reduction. The solution I gave, essentially, was to normalize the powers and compare the bases. In this case, the powers were already normalized, and factorial bases in the range of 100 are obviously higher than 10. In other words, you will obviously need

to go much deeper into 100! than 10 before you exceed its natural base in counting to 100. In other words, counting in factorials is merely an exotic form of counting in exponentials, it's still just counting, albeit a different kind of counting.

Here's the subtle but important point that is at risk of being missed. When we count ordinarily, we do an awful lot of math. First try to understand counting in a foreign base, like base 2, for instance, and then try to imagine how you ever knew how to count at all. It begs the question of what counting actually is. What do we actually do when we count? Everybody already knows how to count. How do we know this? Why is counting so easy and math is so hard, especially since normal counting involves a lot of hard math?

When we count, we are speaking a specific language. It is a language that names things and immediately identifies the logical relationships between the things being named. It is an inherently mathematical language, and it is inherently one-dimensional. In other words, we can always reduce the things being named to a line. The born mathematician does not need to be taught how to count. He not only inherently knows how to count, he knows many different ways to count, and he intuitively knows how to relate the things counted in one way to other things being counted in another. This ability leads to a lot of "free math" to the born mathematician.

The central question posed by this book is this. What happens when we find a "number line" that is inherently not one-dimensional? What does this do to our understanding of counting? What new math does it require to count in this language? What new language of counting does it require? What new math becomes possible with this new language? Who would ever use such a system?

The reason the above anecdotes are useful now is because the concepts are going to repeat throughout the book you are reading. I am a child, a game player, a born mathematical thinker trying to explain to the adults how I do my math, how I play the game that I invented. Life is playing my game the same way in every particular with the way I envisioned it. Once you see it that way, there is no other way to see it. I'm having a helluva time getting people to see it my way.

In essence, I am seeing a world in which there exists a new color. I can see the color, so I know it exists, but I don't even know how to describe it. How do you describe a color? I need to find a person who can describe these kinds of exotic colors, and then I need to figure out a way to describe it to him. It is an exercise in pure abstraction, and that is the stock and trade of a mathematician.

I say I am a born mathematician, and that might or might not be true. Perhaps it is more accurate to say I am a born computer programmer. A mathematician lives at the base of thought – pure abstraction. A computer programmer lives one level above that. He begins with the pure language of mathematics, which is not numbers but logic. The programmer then deals with a proliferation of languages that become layered one upon the next. The point of the languages is purely pragmatic – they exist only to write programs. The programmer then creates worlds and tells stories about how data can become output. The natural progression from mathematician to programmer leads to writer. A writer is merely a computer programmer, but the data and the output are merely the thoughts in one person's head relative to the thoughts of another. He must take the languages in common use and

translate them into a language of his own. He must create a world and the language to do it; then he must populate it with characters and action. One thing I clearly am not; I am not a born biologist.

As a writer I am interested in writing a description of Code World. It is a world entirely of my own making, so it is clearly fiction. However, I not only created the world of Code World, I have lived in it for the past ten years. When I describe this, I want it to be clear that I live in one world and wish it were another. I need to create both worlds and show the difference. That will require a very special language. The juxtaposition of these two worlds happened to me first as a programmer trying to understand math. That is the Code World I need to understand before I can explain Code World to anybody living in the real world. Both of these descriptions are merely my imaginings – to the best of my limited ability.

This book is a description of the conversation I imagine I would have in trying to describe my world to a mathematician. It is a description of a world that must take place through the looking glass, between a mathematical child and a mathematical adult. It must include not only my perception of this new color, but my take on why I see it, why others don't, why they should see it, how it might be seen, and what could be done with it once they do.

I imagined this conversation thusly:

1. What is that toy you've got?

Code World.

2. What is Code World?

It is a number line.

3. What is a number line?

It is a physical embodiment of a set of formal rules derived from formal logic, set theory and group theory, all of which are a part of number theory. It provides first, a language for counting, and second, a system of algebra for creating functions and higher level math.

4. Why would anyone want Code World?

It is a simple tool to help understand life on a molecular level.

5. What is life?

It is a self-organizing system of molecular information. Its primary function is a search for molecules capable of creating more molecular information. At bottom, it is molecules doing math with molecules.

6. If life is molecules doing math with molecules, what is their number line?

Nobody knows. Nobody ever thought to ask.

7. I'm calling BS here… Aren't you trying to tell me that Code World is the number line that life uses to let molecules do math with molecules?

We don't know, seriously, but intuition, common sense and empiric evidence is remarkably supportive. If it's not this, then it's something very similar.

8. Why do you say that?

Well, it's got to be something, right? It can't be nothing, right? If you were going to build life from scratch, wouldn't you start with a number line?

9. I don't know. I'm not sure that makes any sense. I'm not sure that is capable of making any sense. But can you at least tell me why this number line is any different from the one I already have, and have grown so fond of?

Code World physically contains two parts. One part is mathematically isomorphic with DNA and the double helix and the "language" it uses to replicate itself; the other part is mathematically isomorphic with the genetic code. So that's a pretty good start.

These two parts together form a third part, a language of "counting" that is identical to the language life uses to uniquely identify every life form on earth. In other words, life counts all possible forms of life and assigns a number to each actual form of life. This particular number line – Code World - exists in a physical form that would be "understandable" to molecules if they were actually counting.

So now, basically, it seems like even if this idea is a hoax, it's a pretty good one. In fact, it seems to me that it is the best possible hoax. Maybe the reason for that is that it's not a hoax; it's just the best possible explanation for something that previously was thought to need no explanation.

10. What is this "mathematical" basis that you refer to?

Code World is based on pure symmetry, and it can be described with the language of symmetry. So in a basic sense, the same kinds of rules that form a standard number line can be used to form Code World from scratch, and then it too can be used as a number line. It will then be a formal language of counting and have its own rules of algebra. These kinds of rules build one upon the other, and we end up with what we commonly think of as math. In the particular case of counting with Code World there is a foundation for creating a precise mathematical correlation with molecular information in life – as we currently know it - and perhaps we can finally see that there might be a natural algebra to that as well.

11. Is this really consistent with the language of life?

Yes. Life is also based on pure symmetry. DNA uses a language of counting that has its own algebra. The whole thing can be seen as derived from the logical structure of a dodecahedron and its logical relationship to other structures.

12. I thought DNA was the number line of life. Is it not?

No. DNA is a number, not a number line. There is a big difference.

13. What is the difference?

Think about what any number actually is. Numbers aren't real in any real sense; they are human creations. They exist as products of logic. They aren't any one thing; they are many things all at once. The numbers we use in ordinary life are highly complicated and highly powerful things.

Take for instance this number: 3420959112984637

What kind of number is it? Obviously it is a big number, and it is written base 10, since it has ten digits. But what actually is it? First and foremost it is a process. It involves an algorithm for generating a name. We have a language for processing this number into something else – anything else. This same number can easily be seen as a point on a line, an area in a plane or a volume in space. It could be seen as one member of a very large society. The number itself is simply a name given to one thing in a world of things, and that thing can be understood relative to the other things.

DNA is a number written in the language of life. This language is base 4. We have not even begun to understand this language, but we can know it truly is a language and it must have a line, or a systematic way for creating unique names. The existence of a DNA number implies a DNA number line. Imagining this is hard enough, but nothing compared to imagining the translation of these numbers to other things, and imagining the creation of new numbers, and imagining the existence of the number in the first place.

14. How does Code World actually make a number line or a language?

It's a bit complicated, I suppose. Let's use the standard number line as a point of reference. There is an obvious symmetry that runs throughout the standard number line. The symmetry is embedded in the rules that form the line. However, the symmetry is limited; it is not complete. The symmetry finds its focus at zero and extends infinitely in both directions from there. The "language" we use for counting in this system is base 10, but the line itself is base 2. In other words, no matter where you find yourself on the line, you have two choices: greater than or less than. This lends itself quite well to naming "quantities" of things. In this language things are always getting bigger or smaller, and that's nice.

Now compare that to Code World. First, it is base 4. In other words, there are 4 choices, call them up, down, right, left. Second, the symmetry of Code World is complete. In other words, there is no zero, and the symmetry extends infinitely in all directions from anywhere. Messy.

15. You say it is a "counting language." What do you mean by that, and what is being counted?

Once again, it's complicated. It's hard to explain because we humans have no experience counting like this, and when we use our experience with actual counting, it only confuses us. This is the point in the exercise that we need to reject the gravity of "normal" math, and let ourselves drift into a different world entirely.

But let's take the simple example of counting to ten. What does this mean? It means start at zero, and then make ten steps to your right – the positive direction by convention. This is simple for a lot of reasons. First, we have zero as a nice place to start. We know what it means to move to the right. We know what a step is. There is only one shortest possible way to get from zero to ten. Now, realize that all of these things disappear in Code World, but we are still just counting.

16. What is the difference between the limited symmetry of a standard line and the complete symmetry of Code World?

A number line has limited symmetry because it has a center and boundaries. There is an obvious asymmetry at each end. If we could somehow connect negative infinity with positive infinity then perhaps we could form a true "ring" and claim complete symmetry, but so far as I know we cannot actually do that. It is Reals modulus Infinity. Also, the standard number line has three kinds of numbers:

positive, negative and zero. Zero is special and sits in the middle, while positive and negative grow in either direction.

Code World, fortunately, does not have those particular problems. There is no center and there are no boundaries. Everywhere is already logically connected to everywhere else, and there is no beginning or end. There are no special numbers and special places. There is no middle, and the positive and negatives are interlaced. There can be many straight lines between two points. For instance, no matter where you are on Code World, there is more than one way to count to ten.

17. How does this translate into counting?

You can't really "count" things in the sense that we think of as quantities, because directions no longer have the principle of commutation, the thing we rely on for quantities in normal counting. Bummer. The only thing we can really count now are "steps away from," and the sad thing is that the more steps you take "away from," on this line, the more likely that you will actually end up where you started out. A really big number of steps have an equal chance of representing no net movement as it does of any other kind of movement. Strange.

18. Wow, that sounds like a pretty bad way to count.

Perhaps. To each his own. I guess it depends on who you are and what you are counting. This counting world is a different world entirely. We are not small molecules, but if we were, how would we count?

19. Funny how this all sounds like Alice in Wonderland. Wasn't Lewis Carroll a mathematician?

Yes.

20. What is this about quantities and "steps away from?"

In normal counting we have a clear sense of what it means to “add one” or to “subtract one” that correlates to physical locations on a line. In Code World that sense betrays us, because we can never actually travel in a “straight line.” What it really means to “add one” is to perform a transformation, or to transpose the current status to a new status. In Code World it is all about transformation and transposition, not quantification. The only quantification in Code World is in the quantity of transformations.

The basic problem is this: The relationship between right and left is always preserved on a standard number line. This makes it quite handy to quantify things in the real world. In Code World, all relative values must be re-calibrated after every move, so it makes it impossible to “quantify” real world things with this kind of counting. It is still just counting, but the counting must take on a new meaning. In other words, a molecule never asks “how many?” Molecules always ask “where am I,” and “where do you suppose I should be?” Basically, when we count, we count members of a set. In Code World, the members of the set are themselves sets whose logical relationships involves members of other sets, so the members of the set being counted are in the set of logical relationships of the things we normally count. So it’s like normal counting at least one level removed.

21. Can you give me a real world example of how this difference effects our counting?

Okay. Think of an ordinary number line. If I tell you to “add one” you already know what that means. You actually have a choice though - add one what? You could add one positive or you could add one negative. This is predicated on the innate sense of what “one” is and how we might add it. One is a single transposition on the line. Now if I would tell you to add one positive and one positive, you would know exactly what to do. 1 + 1 = 2. If you followed my instructions, you would be standing on 2. This logic does not apply to Code World. I tell you to “add one” and you ask, add one what? Now there are four options, up, down, right and left. Okay, I say add one up. Now you know where you are, you are one above where you were. Now I tell you to add one up again. Now where are you? Obviously you are two above where you started.

Nope.

You are back where you started. On the ordinary line if I tell you to add one positive, add one negative, add one positive, add one negative, you end up where you started. But in Code World if I tell you to add one up, one down, one up, one down, you do not end up in the same place; you end up four transpositions away.

22. How is that possible?

Like I said, it's a different world with new rules. In Code World, up + up equals null set, and up + down + up + down equals many possible things. Perhaps another way to look at it is that after each move there are three possible positive moves and one possible negative move, a move that negates the last one. With a standard number line the destination is important and the journey less so. With Code World the journey is important and the destination less so. It's not where you are, it's what you are relative to.

23. What exactly are these "normal rules" of counting that you keep talking about?

I'm not entirely sure (I don't think anybody really is) but whenever we usually count, what we actually count are "transformations." In the normal sense, what we are transforming is a very well behaved position on a number line. So the number 126 represents a specific quantity of net transformations to the positive end giving us an easily quantifiable total transposition from zero to 126. It is quite easy to keep track of where we are and what we are doing and where we are going. There might be many possible "oscillating" or "cumulative" ways to get there, but there is only one shortest possible way to get there. When we count in Code World we are still counting transformations, but they are literally transpositions of one shape on another. It correlates with a literal description of two shapes changing their relative positions in time, so it would be equally valid to say either shape is moving relative to the other. And so we can translate these transpositions from the vocabulary of one shape to the other.

24. What precisely is a position, and what then is a transposition?

If we take the reference shape to be a dodecahedron, then a position might be seen as a tetrahedron relative to that dodecahedron. A transposition is then a changing of the position of the tetrahedron via a sequence of transformations. We could easily reverse the relative roles of the two shapes. The tetrahedron could be the reference and the dodecahedron could be the transposed partner. But we can also use either as the basis of the vocabulary directing the transformation. There are many "possible languages" that would achieve the same effect. For instance, instead of a tetrahedron, we could use a cube. A cube is a dual tetrahedron. So instead of speaking in terms of a single tetrahedron, we could speak in terms of a positive and a negative tetrahedron. The net effect is identical.

So the “language” of Code World is set and totally objective before we come along and play with it. It is merely our subjective preference as to how we speak it. We could just as easily speak Code World in any of the perfect solids as in any other. For that matter, we could use an incredibly large number of polyhedrons to “speak Code World.” The structure of the language is objective and broad, but any use of it is highly idiosyncratic.

However, as a practical matter, we must always assign Code World three different vocabularies. I called them Globe and Glider and Code. The task at hand is to specify two positions and then find a sequence of transpositions between them. Globe is usually spoken in dodecahedron (or if you prefer, icosahedron) and glider is usually spoken in tetrahedron (or if you prefer, cube or octahedron). The sequence of transpositions is optimized in tetrahedron. Any of them could be spoken in any of the others. It is a matter of preference by the speaker. You could use dodecahedron to speak dodecahedron on a dodecahedron, or in the simplest case, I think, you can use tetrahedron to speak tetrahedron on a dodecahedron. The best way, I think, is to use a cube to speak tetrahedron on a dodecahedron, but that toy is a little harder to build.

So the game could be played in a huge variety of ways. There is a natural algebra that works like this: C(A) = B, where A, B and C might stand for many things depending on how we decided to play the game. A could be a dodecahedron, C could be a tetrahedron and B could be a new dodecahedron.

You can perceive it however you like, but in the most abstract form it is a language that uses the interaction of shapes to form new shapes. The language is used on a well-defined small set of shapes, taking advantage of the fact that they are all subsets of each other, but when it is played over time and recorded in some way, it represents a large number of large and complex shapes.

25. And you think this is the language life is using?

I don’t know, but it seems like a fabulous place to start. If life is not using this exact language, it must be using something that is extremely similar. My theory is that it must be using some language, and the language it is using must precede the way it is being used; otherwise, how do you play and record the game over time?

What it is, at bottom, is a language of polyhedrons by polyhedrons for polyhedrons. I believe that life uses, at bottom, a language of molecules by molecules for molecules. It seems logical to me that this is that language. Life is molecules doing math with molecules.

26. This sounds like an awfully complex and clever language, and you don't strike me as being particularly smart. How did you invent it?

Thank you. Finally. Now you get it. I did not invent it. I am not smart enough to invent such a thing. I discovered it. I invented a device that uses the language. I merely searched for clever symbols that would allow me to use the device to speak the language. The language existed in the shapes before I came along.

I readily concede that I am not very smart, but I am at least as smart as any bag of small molecules. If I couldn't invent it, then neither could they. My conclusion is that they found it just like I did. The language must have already been there for them to find. More importantly, I concluded that some language must have been there first, because life must first "record" its information before it can refine and expand it. How do you record information if there is no language in which to record it?

27. What is this stuff about "speaking dodecahedron" as opposed to some other shape?

In a real sense a dodecahedron is merely a collection of parts – faces, edges and vertices. In a mathematical sense a dodecahedron is a collection of kinds of information and a logical relationship between them. In other words, if we know the logical relationships, we only need to know a small part of the information to derive the rest. We could use any small logical collection of the parts to provide the full amount of information. It's a language of shape compression within shapes.

If we focus on the vertices as the "seed" information, as I did initially with building the game, then a dodecahedron is very useful for generating "simpler" forms. In other words, there are lots of cubes and tetrahedrons already in it. Plus, there are simple logical rules of transformation that we might use to project them into icosahedrons and octahedrons, and all combinations thereof. So, the parts of a dodecahedron provide the most efficient way to translate into variations of a dodecahedron and into variations of all the other solids. The "language" that develops derives from the logical relationships between the subsets. In a very real sense, this is what Code World is. The actual execution of this

language depends on which parts of which sets we chose to use. We could chose to use any parts of any sets, but we are interested in the "best possible" language.

So there is really only one language, if you want to look at it that way, but there are a large number of ways it could be spoken. Life is actively speaking this language and actively looking for the best possible way to do it. DNA speaks cube when it replicates itself. It speaks dodecahedron when it translates into protein. Protein is based in tetrahedron, but lord knows what it's actually speaking when it translates back into more DNA. DNA is simple when isolated, but protein is never simple. We should never pretend that it is. The important point is that you can never isolate DNA from protein. If you do, the language itself disappears.

28. Are you actually able to play this game with this language?

Sure.

29. Are you any good at it?

Not really. I thought I would be, but I was wrong.

30. And so your specific claim about Code World and life is what?

Code World is a physical embodiment of logical rules that represent a language of shapes. These rules are evident in the molecules of life today, and logically preceded all life on the planet. In other words, Code World is a physical embodiment of what easily could be thought of as the first life form on earth, and so Code World provides the best clues to the origin of life on earth.

31. You think Code World represents the origin of life on earth?

Sure.

But if not Code World, then we should immediately and aggressively begin a search for the best alternative, because logically we can assume that it is something very much like this.

32. That sounds mighty salty, but what practical use is such an idea, and why are you so sure that such a thing must actually exist?

Well, I suppose what we have here is called a hypothesis. Call it The Language First Hypothesis. Unfortunately, what is currently accepted is the negation of that hypothesis. I think we should test it… at least. What's the point of a hypothesis if you don't plan to test it?

This particular hypothesis, as far as I know, was never formally stated. The negation of it has definitely been formally accepted. There are many detrimental consequences to accepting the negation of this hypothesis – especially without testing it. Yet I believe that common sense demands rejection of the negation of the hypothesis.

In fact, I feel like the actual negation of the hypothesis is logically impossible. In other words, I think that logic demands we actually accept the hypothesis.

The point of my fiction is to imagine a world where the hypothesis had actually been proposed before it got rejected. Had it actually been first accepted, we would be imagining a radically different world. I think we should first recognize the world in which we are living, and then imagine the world that was first rejected. We could then compare the two worlds and see the difference. Therein lays the value of the idea… since you asked.

33. What, in your opinion, is the nature of the world in which we live?

We live in a world where art apparently imitates life, in the sense that it is the art of science imitating our discovery of life. In the actual world, discovery preceded theory, and so the conclusion was that no theory was needed. Unfortunately, this is the same as saying that no theory is possible. In other words, if no theory is needed, all theories are equal, so there is no possible way to argue for one theory over another.

34. What was the discovery?

We discovered sets of molecules, and crude relationships between the sets. There was no real theory predicting what these sets might be or what their relationships might represent; therefore, it was assumed and later explicitly stated that no real theory was needed, ergo no theory is possible. In other words, my team got shut out even before we got to the plate.

35. You are starting to make me a little nervous. You seem to do a lot of ranting, for one thing, and you've mentioned the Bible, you've admitted that you are not an expert and that the experts don't agree with you - universally. Not to put too fine a point on it, but you sound like a heretic.

Is there a question in there?

36. Do you know why you might sound like a heretic?

Yes. I am a heretic.

37. Would you care to elaborate?

Yes, thank you. People can be classified by their priorities between knowledge and belief. We tend to think that religious people value belief over knowledge and scientific people value the opposite. I have not found this to be true; I think all people value belief over all else. It is when we get to the next level of priorities that the nut cutting is done. There are those that believe they know, and those that believe that they don't, and I think these people are spread fairly evenly over all spectrums. When it comes to the genetic code, the scientific community is unified in their belief that they know it. I am quite certain that I don't know the genetic code. I am a heretic because I am challenging the belief system of others who do indeed believe quite strongly. It's not always pretty, this whole heretic business.

38. So you are saying that the scientific view of the genetic code represents a belief system and not knowledge.

Yes. Absolutely. Call them helix worshippers. It is a very strong belief system based on limited and flawed knowledge. The belief has become more important than any actual knowledge about what they believe. The actual knowledge produced by this belief system is now producing gibberish on a quantum level, and it is being spouted in a language that is faulty to the point of being self-contradictory and logically impossible.

39. So you don't believe in the genetic code? You don't believe that codons translate into proteins, despite mountains of data that prove that they do?

No. That's not even close to what I just said.

40. What do you believe?

I believe that we have excellent data. We are already doing fabulous things with our data. But I think we are interpreting it improperly. I don't believe that we know what the genetic code is, how it works, what it means or how it came into being in the first place. I think huge liberties are being taken that are wholly unwarranted and incorrect. I think the net effect is to lead us in wrong directions and cloud our ability to obtain a better understanding in the future.

41. So you think that by not understanding the genetic code you understand it better than all of the experts?

Exactly.

42. Besides that seemingly stupid point, can you give a specific example of your disagreement with the scientific community?

I'll give it my best shot, but there are so many points of disagreement. I suppose the simplest way to state it is in a test of the simplest form of my general theory. Imagine if everybody had to take a test and it contained the following question:

The genetic code is a set of molecules executing a language. Which of the following is true?

A. The language preceded the molecules.
B. The molecules preceded the language.
C. Both.
D. Neither.

There are really only two answers: A and Not A, because B, C, and D are now logically equivalent. C basically means that the molecules and language evolved together. D basically means that the question is invalid for any number of trivial reasons, so any possible answer is meaningless. B is the implied answer merely because this is the way it actually happened in discovery: We discovered the molecules and built our view of the language around it. This is logically impossible and epistemically worthless, in my opinion, so now everybody just defaults to C or D. There is a very strong belief in Not A, that's for sure. It is virtually universal, and it is sad to watch otherwise intelligent people twist themselves into a pretzel trying to defend it.

43. But you know better?

Yes. I know that A is not only logically correct, it is epistemically quite valuable.

44. Can you explain how you know this?

Not very well.

45. Can you try?

If you insist.

I don't have "experimental" evidence of the kind being demanded of me, but we can do any number of thought experiments and conclude its truth. More importantly, I think we should recognize all the

things we could do with a belief in A, if we actually had one. We need to realize that Not A is the wrong answer.

Let's start by assuming we know pretty much everything about everything and nothing about life. The question we want to answer is simple: What is life?

Since we already know everything, we know how to time travel. So let's use that ability to collect data on the history of earth. We get our best man, call him Rex, and we stuff him into a time machine to collect data on earth. What do we want to know about earth? Everything, and let's say we want to know it at fairly close intervals, say every million years.

Rex sits in his little time machine and carves out a section of the universe that can reasonably be called earth, and this section remains constant throughout the period of observation. He has a measurement device that has the ability to capture "all possible" information about earth, call it Liz. It divides down the space that earth occupies to the absolute smallest spatial unit, and within each of those units it assigns either a zero or a one that represents the state of that unit at that time. As a control he does the same for the other planets in the general vicinity. The raw data is collected in an instant, and now it needs to be stored. Like any good digital device, Rex's instrument contains a compression algorithm, call it Ralph. In fact, it is the best possible compression algorithm. It is lossless and achieves maximum compression.

After billions of years of collecting this data, Rex notices something odd about earth compared to the other planets.

46. What is it?

Great question!

Here's what I think it is, and I think it gives us the best possible clue to the primary question at hand. I believe that the number of disks required to store earth's information is increasing at an exponential rate.

The only significant difference between earth and the other planets is that earth has life, so Rex crudely assumes that life is somehow seen by Ralph as an exponential increase in information. But the increase in information, as far as Rex is concerned, is solely due to his compression algorithm. In other words, without the algorithm there is no increase in information. Unless the data is being compressed, Rex is unable to differentiate Earth from Mars.

Logically, then, as far as Rex is concerned, life is merely part of a compression algorithm. Any other question Rex might now ask of life can equally well be asked of his algorithm. As far as Rex is concerned, understanding life is the same as understanding his algorithm. Not to put too fine a point on it, but, Ralph = Life.

47. That's it? That's your logical proof, your claim that life is a compression algorithm?

Yes. What's wrong with it?

48. Well, first off, it sounds pretty stupid. Plus, wouldn't your detractors just claim that "the genetic code" as they present it is also an algorithm?

Precisely!

They present it as an algorithm, but the nature of their supposed algorithm is decidedly shabby, and they make no attempt to fit it into the larger algorithm of life. They somehow think that life is more interested in storing information in the least effective way. When it comes to raw data and the genetic code they insist that Ralph will decompress it for storage. When they try to explain why they think that this is logically a good idea, they must conclude that they are spouting self-contradictory gibberish.

That's usually when they say they don't want to talk about it.

49. And?

Back to my basic point, which is this: We don't know the actual algorithm that we love to call the genetic code. We aren't even close to knowing it. We might never know it. We aren't really even trying to know it. But perhaps we can know the general nature of it. We can get a much better general sense of what it is, how it works and what it is doing, but only if we concede that we never really knew it in the first place.

50. So that translates into what specific real world problem?

Well, as far as I'm concerned, announcing that we cracked the genetic code was the largest false alarm in the history of science. At best we cracked only half of it, and we started with the wrong half. It was like placing a steel ball on a wedge, and it started rolling down the wrong side of the wedge, which led our thoughts and words in exactly the wrong direction. My fiction is based on the supposition that the ball easily could've - and should've - fallen in the other direction. Unfortunately it did not, so now there is a lot of unfortunate work to be done in getting the ball back up to the top of the wedge so that we can let it fall in the right direction this time. There's a lot of bad language to hack out before we can start using the really good stuff.

51. Since you aren't smart and they are, how do you know this and they don't?

At bottom it is because I know how to make an algorithm, and I know the value of starting with the algorithm, its function and structure and abilities. I know what it takes. And totally coincidentally I had to make the exact same algorithm, in some fundamental way.

52. So you had to make the genetic code?

No. But in a very abstract way I had to construct my own algorithm from the exact same first principles as the algorithm that became the genetic code in the world of real molecules. I knew what I needed. I constructed it. Life needed the same things. It constructed it too. Only the one it constructed is infinitely more complex – almost perfect, really - than any I could ever construct. It is doing things that we can't even imagine are possible, and we aren't even trying to imagine them. The problem is not so much in a failure to recognize correct solutions to problems; we are utterly failing to recognize the problems.

53. Why do you suppose that is true?

Obviously, it is because of the strong belief that we already have correct solutions. However, even a cursory examination of those solutions reveals they are incorrect, yet we happily go on ignoring the true problems.

54. Let's just suppose for now that your stupid main claim is correct - the simplest key to understanding life boils down to understanding Rex and his algorithm. How do you propose to proceed and how will that help our understanding?

I believe that at this point in time, our ability to understand life depends on our ability to reverse engineer this particular algorithm. Call it blind faith. But let's suppose it is true, in some sense.

The first task is to assess our ability to reverse engineer this algorithm. I believe that we have no chance whatsoever. I believe that it is so complex as to defy human comprehension. That does not mean that we cannot begin trying to understand how we might understand it.

55. So what is the first step in that process?

Unfortunately, the first step is that we have to ask basic questions, and we have no way of giving answers, other than a wild ass guess.

56. Give me an example.

Okay. The most important question is this: Is the algorithm symmetrical? In other words, is the "best possible" algorithm the same no matter where you are in time and space, or are there different algorithms that work better depending on when and where they are applied? I don't know how to go about answering that question, but it must be answered. After all, we need to decide if we are making one or more than one algorithms.

My intuition - my belief - is that it is perfectly symmetrical. In other words, the most efficient algorithm for describing Mars a billion years ago is the same most efficient algorithm for describing Earth today, or earth three billion years ago.

57. What is your basis for this belief?

None whatsoever; call it blind faith if you must. If nothing else it allows me to move on and try to answer other questions, happy in the knowledge that I am trying to understand a single algorithm and not many different algorithms that change in time and space.

58. Okay, I'll give you that for the sake of argument. But I see a much larger problem in everything you are saying. Your basic premise might be fatally flawed. Let me give you a simple example. Let's say that Rex is able to capture data on a particular segment of the universe, let's say Mars three billion years ago. Your premise is that this is perfect data. In other words, Mars three billion years ago can be captured, stored, and recreated perfectly. Given that - and that's a lot to give - my question to you is this: How could this data be stored in any form that is smaller than the actual data? In other words, the process of "compressing" and storing the data must surely involve some loss of data. But if the algorithm is "lossless" then where does this information go if it is compressed? In other words, can you actually store a universe on anything smaller than that universe?

That is an excellent question. I've got to hand it to you. I didn't think you were that smart or paying that close attention.

59. Answer?

Well, once again, we must make a decision, and we have nothing to go on. Either the data is compressible or it is not. If it is not, then we are done here - there is nothing left to talk about. Based on the strength of that logic alone, I'm going with the answer that it is indeed compressible. I don't know how or why, but now that we have made that decision, we can move on to the next obvious question…

60. Where does the information go?

Yes. Good question. The answer is that the information is contained in the algorithm, and the algorithm must also be in the universe. After all, what is a universe if it doesn't contain everything? There is no other logical answer. The algorithm must contain descriptions of common, repeatable patterns in time and space, so in order to compress information, an algorithm must first contain information. The more information it is able to contain, the more consistency of the patterns in the data, the greater its power to compress.

61. But you said it was only one algorithm. If patterns evolve over time in the universe does the information in the algorithm have to evolve with it, and is this information counted against the number of disks Rex must use to record his compressed information?

Yes and yes. The algorithm will go on the disk with the data. Presumably as the size of the algorithm gets bigger, its ability to compress the data will get bigger too. However, the actual compressed file will get bigger too, which reflects an exponential growth in information.

62. Okay, how would you do this trick with this algorithm that doesn't sound to me like it could ever exist?

Well, I would start by trying to think of all the essential elements?

63. Okay, what are they?

If I were the guy trying to make the algorithm for Rex, the three things I would need at first would be space, time and language.

64. Okay, I understand space, but I don't understand time and language. Can you explain?

Maybe. I'm not really sure. Space appears to be the easiest… or maybe not. It is pretty intuitive. Time? Not sure. But let's think of the simplest form of language that is familiar to us in the parlance of our day. Let's say I come to you and say that "I have three cats." According to Claude Shannon, we are

communicating information. This could be simple or it could be complicated. Implied in the system are three things: sender, receiver and message.

If the only message ever communicated in this simple system is the number of cats I have, we have a pretty easy process. All we need is a language that "counts" and then uses a code that transmits the number that is counted. Unfortunately, it is never this simple. If I am sender and you are receiver, then the range of all possible thoughts I might have is a set. The range of all possible thoughts you might have is another set. Those are the sender and receiver. What is the message? It is two things: information and language. The language is embodied in all possible correlations between sender and receiver, and the "information" is contained in any one precise correlation between the two.

So let's say our simplest possible system gets just a wee bit more complicated. Let's say that I want to include information about dogs. Let's say I want to include information about not just me but you as well. Let's say I want to include relationships that include not just "have" but "gave," "got" and "borrowed" as well. Let's say I wanted for you to have the ability to not only receive the message but to send one back as well.

Now we can understand that this "language" we are using, as simple as it might be, will need the ability to not only transmit but organize information as well. It must have ways to keep straight the relationships between sender and receiver, and substitute one for the other. It must have the ability to not only transmit information, but to contain information. On top of all this, it must have some fundamental way to "count" because at bottom all information depends on counting. So now the load for transmitting information is rapidly being shifted from the sender and receiver onto the language itself. Of course the sender and receiver must also have a way to store and interpret the language. The languages that are best suited for these things are the ones that are most compatible with the sender and receiver, and the ones that are most efficient at doing all of these things.

Here's my main question: Can there ever be information without language?

65. Who knows. Your point being?

In a world of molecular information, what could possibly be the sender, receiver and language, let alone the message? More importantly, what could the thing be that we are calling molecular information in the first place? Could this thing ever possibly exist without a language?

If it can, I am at a total loss for how that would work. There must be some way to correlate the "information" between sender and receiver. Without that, what is information?

66. And that has what to do with Ralph?

Good point. So now we need to imagine the task at hand with Ralph. The "sender" is the data that Rex captures. The receiver is the data that Rex stores. The language is the way that the algorithm organizes and transmits the information from sender to receiver. The algorithm is nothing but a set of rules for doing this. These rules must not only contain logical relationships, but methods of counting and storing units of information.

67. What might the rules be?

We don't know, but we can imagine.

Let's suppose that the universe is such that it forms patterns. Let's suppose we can call one of these patterns "hydrogen." What is hydrogen? Who knows. But let's say that it is composed of sub-patterns, call them proton, neutron and electron. What are these? Who knows. Surely they are a single thing in some sense. Let's say we could somehow describe every possible thing that might be an electron. The problem is that on the scale that Rex is working with, there are an enormous number of possible things that can be called "electron." Let's say that every possible thing that could be called electron is somehow logically related to every other possible thing that can be called electron. Let's say that all of these things are logically related as a function of time. If this were true, then all Rex's algorithm – we named him Ralph - would need to do is say "electron at T_n" and all of the information would be captured. If this were also true for neutron and proton, and perhaps hydrogen, then Rex would have a toehold on compressing his raw data. Even though all possible descriptions would be unfathomably enormous, this information would only need to be described once. Of course it would need to be captured in the algorithm itself, and we have yet to see how that is possible, but the efficiency of data storage would be staggering. The actual amount of information contained in Rex's raw data for any hydrogen defies our comprehension. As simple as hydrogen may or may not be, it always represents an unmanageable amount of information at any given place or time, at least it does at the level our brains are able to conceive it.

Let's just suppose this is possible…

68. Excuse me, but that seems to be nothing more than an idiotic description of basic chemistry. Are you talking about the periodic table of the elements?

Yes, precisely my point. Chemistry is based on the quantum model of the atom. This represents a compression and digitization of discrete chunks of the things that Rex collects as data. Because these things are predictable in time and space, to some extent, we can create various forms of shorthand for them. What we find is that it is most effective to view them as collections of parts, and those parts are also collections of parts. It is somewhat of an atomic self-organizational system.

It is also somewhat ironic that something that actually turns out to be as "big" as hydrogen would first be called an atom, since atom is meant to be something that is indivisible. So now we speak comfortably with cool oxymoron words, like sub-atomic, which is like cold water heater. Strike that - it probably should be called cold water heater, according to George Carlin.

69. So in terms of "compression algorithm" what you are talking about are things like the periodic table of elements.

Yes. Exactly. The periodic table is what we might call a coordinate system for information. It is a graphical representation of a formal set of rules and it organizes information to help us understand it. We can use those rules to not only understand the information, but also to compress it when it is in use by us to do chemistry. It is the blueprint for how these things make other things - like molecules and crystals, for instance.

70. Ralph will contain a periodic table of the elements?

Absolutely, as well as many tables above and below that, no doubt. But Ralph will use a single coordinate system that ties all of them together under one roof. This is the essence of my belief that language preceded molecules. Language at this level must precede all of the parts that make molecules; that's how molecules get made.

71. Who are you to know this?

That is an excellent question, and an altogether fair question. I realize that I am expounding an utterly fantastic theory, but it is based entirely on a belief. My belief – which at this point is unshakable – is based on an experience, and an experience is entirely subjective. In order to have that experience you must be the one having it.

If I describe myself a wee bit, not only will you have a better understanding of the experience, but you will have a better understanding of what I am doing with this theory and why I need help doing it. You can see why a writer of fiction, inventor of toys and a constructer of universes might need a man of skills mathematical, such as yourself.

72. So you are going to tell me your life story?

Well, no, only part of it, hopefully only the important parts.

73. If you must.

When it comes to any kind of formal education, it is entirely appropriate to call me a slacker. I resisted being trained to the extent possible. The first question I always asked in any class was this: What is the least I can get away with doing in this class? I say this only because some classes will compress better than others.

The sequence in which they actually trained me was this: P-Chem (just teach me the rules for making an atom and I will fill in the blanks - Minerology (just teach me the rules of forming a crystal and I will fill in the blanks) - Bio-chem (just teach me the rules of forming a bio-molecule and I will fill in the blanks). The reason I mention this is two-fold. First, I always travel light when it comes to knowledge in formal academics. These are the subjects that traditionally cause the most trouble, but ironically they can be traveled the lightest.

At the end of my formal training I proudly considered myself to be almost information-free. You don't need to know very much, but with a little bit of knowledge you can answer an awful lot of multiple choice questions and fake your way through it. Second, I learned these things in exactly that sequence, and I felt that you could merely start with a little bit of knowledge and only need to add a little bit more to get to the next level. So you might call this a cosmology of test taking, if nothing else.

Understanding chemistry allowed me to bridge the gap with minerology – I saw it as the way atoms pushed other atoms around to form crystals. When I made the leap to bio-chem it was a whole new set of rules, and the new rules broke the old ones. No problem, just give me the rules and the test and tell me where the kegger will be afterwards.

Now I understand that the rules can be internally consistent, if you know how to understand rules.

74. Anything else you think I need to know about you?

Yes, to belabor the point yet further:

I am an individual who cannot remember how to spell his own name. That is not a joke. (I can explain.) I also vividly remember a particularly ugly argument with my second grade teacher. My position was that the day I was out they changed the alphabet. Her position was that they did not. I think I won that one, but at the very least she now knows her place.

75. There is a point in here?

Hold on, I'm just getting started.

I entered medical school with a personal lifetime goal of never touching a computer. I barely had the ability to format and punctuate even a simple English sentence. It is merely a testament to the power of standardized test taking in American education circa 1980.

The turning point in my life, as far as I'm concerned, is when I started to teach myself how to program a computer. In order to do this I felt like I needed to understand the computer at the processor level, which is not a good place to start in general. However, it was one of the most fulfilling and liberating experiences of my entire life. On the basis of that I actually learned English. It was only through programming that I finally understood the basic concept of any written language. By comparison, English is easier and less creative. Programming a computer is harder, but far more creative. It is much more rewarding, if you ask me.

Eventually I started a software company in which I programmed my own little universe. I now recognize that I have a mild and unusual form of autism, whatever that is. This was the world in which I could happily live.

My programming style, as you might expect, is a little eccentric. I believe that the primary task of writing any program is contained in defining the data. This philosophy perhaps might be familiar in what are called object oriented languages. What this basically does is blur the lines between data and algorithm.

76. Is there even a minor point on our horizon?

Yes. From this ontogeny of thought I have naturally come to believe that the vast majority of the data in life resides in the algorithm. The result of ignoring the algorithm is that we miss virtually all the relevant information, not to mention ALL of the relevant algorithm.

More specifically, this is the general nature of the mind that is spewing these bizarre ideas over these many pages. I am not an idiot (in the traditional sense of the word) and I am not a savant, but I am some kind of idiot savant when it comes to the idea of Code World. And what I ultimately came to "know" as true about it, comes from a specific experience I had with it.

77. Is there any way you might relate this "knowledge" in a way others might understand?

Well, I suppose it is a lot like the experience of seeing a new color. If you had that experience, how would you know it, and how would you describe it? How would you know it was a "new" color, and how would you be able to get other people to "see" it the way you do?

78. And your point relative to Code World is what?

My experience was the equivalent of seeing a new color, and it involved two discoveries that were unique to my circumstance and my way of thinking.

79. Okay. What were the discoveries?

The first discovery was Code World.

80. And the second?

I discovered that life had discovered it first.

81. And that's how you "knew" it had meaning?

Yes, because I knew from long experience the methods and reasons and benefits to discovering Code World, and I knew that these were the same for life.

82. Can you step me through some of those things?

Okay. Perhaps I should say that the first discovery involved the general process that led me toward the discovery of Code World, which was this. If you want to describe anything in space and time, first you need to have a definition of space and time. It would seem that space is easier than time, but actually I think it is the opposite. To have time you only need two things: before and after. I think you can fudge everything else. Maybe not. Let's say that's true. Let's say you want to do space the exact same way as time. You can do that too, but what do you have? You have a line. What if you want more than a line?

83. And the answer is?

Ironically, as a programmer there is only one thing available: a line. All data must be reduced to a "string" which is just literally a line of zeroes and ones. In this way, every chunk of code, every algorithm, every program, every whole computer can be seen as a single number, one long number with only two digits – 0 and 1. But this begs the question of what is a line in the first place. As a programmer a line is very literal: it is a sequence of bits. Here's the problem: a sequence of bits is not the same as a line in physical space. But the disks that contain Ralph and his work are literally a single line of zeroes and ones – as we are able to understand them. So a programmer's fundamental task is to figure out how to make all things using only lines.

Now we are forced to make real decisions. There are lots of different ways to frame these decisions, but the bottom line is this: If we want to make a representation of space and time from our data, and our data must be a single sequence of zeroes and ones, how do we convert this literal sequence of zeroes and ones into any kind of representation of a "3D" universe? Is the universe really 3D? We need to figure out ways of chopping up the linear universe into smaller bits of zeroes and ones and define how to put them all back together. One would hope that the actual universe contains this definition somewhere in the fabric of the universe.

There seems to be only one way to start.

84. And that is?

Dice the sucker up into cubes.

85. So that's what you did?

Of course. To me, it was programming 101. I needed to define space, address space, and assign information to space. Essentially, the only choice was a Rubik's Cube.

86. So what's the problem?

These tasks are terribly simple, to be sure, but the implications of doing it the only way I knew how were ugly. The first question is, how many ways, really, are there to do it? The next question is, are they all equivalent? I didn't have any clue how many ways there were, but I intuitively felt they were all equivalent. Intuitively, I felt that I was limited to cubes. That was the easiest way to see it, and I couldn't imagine any other way to do it. So, basically, I felt pretty sure I needed to start with a standard orthogonal system of data addresses.

Of course there are many and various advantages to this system, starting with the fact that it is the basis of everything we know. It not only defines the way we see everything, it defines the way we talk about everything and translate everything into something that we do know. In short, it is how we count and how we talk.

87. Is it possible that there is a way to do this that is simpler, better, or not logically reducible to this?

What I discovered, quite accidentally, is the answer is definitely yes.

88. Can you explain how you might do this?

The orthogonal system is founded in our system of geometry. Specifically, we define a point, two points define a line, and three points define a plane. Any coordinate system consistent with simple geometry, theoretically, I should think, would logically reduce to this. In other words, an alternate system that does not logically reduce to this would be inconsistent with geometry. Right? Maybe. If you found a coordinate system that doesn't reduce to this, what would you conclude?

Well, that was what I discovered. The most striking thing about it was how simple it was. The only problem, it seemed to me, was that we have to redefine our system of geometry. No biggie.

The first problem with an orthogonal coordinate system – perhaps not a problem but a reality – is that no matter how I conceived of having information "located" within the system, they were logically equivalent. In other words, I could image the information being located at a point, edge face or

volumetric center, and conceptually it didn't change. This leads to a lot of basic problems. First of all, if you want build an information system that changes with time, you need to do two things. You need to store the information, and you need to propagate the information through it, or you want to "move" information from one place to the next. With an orthogonal system the information always is a cube and it always propagates as a cube. What if you want something other than a cube? Second, What is a line? What are all possible lines? They are always parts of consecutive cubes. What if you want lines that don't fall exactly on these pieces? Now think about how you can answer these questions without a lot of programming. What if the programming were to be minimal and the questions could somehow answer themselves? What if the coordinate system itself could somehow contain answers?

89. What are the answers?

These are, understandably, not easy questions. I'm not sure I have any answers. But let's take the question of propagation, or getting the information to move. Even if you can't move the information the way you like, perhaps you can somehow move the coordinate system. Why must the coordinate system be fixed and the data moves within it? Who knows.

90. The ways to do this are what?

You either rotate the coordinate system through time, or you build multiple coordinate systems and move the data between them.

91. Why is this any better?

I don't know that it is. But the point is that you are either going to have to build a coordinate system and program the desired features into it, or you are going to have to build a coordinate system that already has the desired features.

92. Do you think this can be done?

I think it can. I think there is already such a system, and there is a name for this system. It is called a dodecahedron. If your coordinate system is a dodecahedron instead of a cube, a lot of crazy things start happening.

93. What are these crazy things?

The craziest thing, in my opinion, is that it will not logically reduce to a cube. In other words, you can compress a cube into a dodecahedron, but you cannot even create a dodecahedron from a cube. This seemed pretty crazy to me, but it seemed to have a lot of favorable consequences for what I was thinking about doing, or more accurately, what I was thinking might be done. Unfortunately, it utterly destroyed my understanding of geometry. No biggie, I'll just have my own understanding of geometry.

94. Why does your understanding of geometry have to change?

The first thing that I noticed was that you couldn't make a "straight" line in the system. I should say that you can make plenty of straight lines within a dodecahedron, but not between them. I actually saw this as a feature, not a bug. I wanted to imagine what light looked like when it traveled through the universe. If it travels in a straight line, how do you model it with a cube? It seems like you can only have three "pure" lines, and the rest of them must be the result of a programming trick. With the dodecahedron it felt like there might be a possible solution that is simpler. If light is seen as the propagation of information, then if it propagates between dodecahedrons I thought I would somehow naturally have more straight lines.

95. How could you get this done?

That's when I invented Code World. It is a simple counting device. It counts the number of ways to define the dodecahedron using another shape to do the counting. In other words, information can be stored as a dodecahedron and propagate as a tetrahedron.

I thought this might be epistemically useful, and I knew that it would expand my ability create a lively coordinate system, but I also thought it might make a pretty cool game. That's when I built a physical model. That is Code World. I needed to see it first before I could make it actually do anything.

96. Then what did you do?

Well, that's a pretty good question. I'm not exactly sure what I did. The first question is how do you actually address information within this system? Theoretically, this seems pretty easy, but in reality it is not as easy as it sounds. That are lots and lots of problems that didn't even occur to me at the time. First, how would a computer do it? Second, how would a human do it, at least how would a human do it in a way that makes it possible to play a game? I knew the two questions were different, but I didn't fully understand the implications of the differences at the time. I was entirely ignorant of the salient problems. Naturally, I started working on the human version first, since I am a human. Even here it turns out that there are an enormous number of ways to possibly do it, so I decided to find the "best" way to do it for a human first. That is Code World as I first saw it.

97. How did you start?

There are three pieces to this particular puzzle: Two coordinate systems and a logical way to relate the two. The two systems can be the same or different. The relationship between the two could be based on many different things.

I tried a lot of things. What I settled on was a "fixed" coordinate system containing the full information of a single dodecahedron based on a single description of its points, or more efficiently its planes. The symbols I finally chose for this included two shapes and five colors. Each color represented a cube, and each shape represented a tetrahedron in that cube. The second coordinate system I chose was a maximal compression of the first – a tetrahedron – based on four symbols, one for each point. The relationship between these two could then merely be stated by calling the single stationary point when the tetrahedron transposed on the dodecahedron. This was the "simplest" and so I decided that it must be the "best?"

98. How did that go?

The whole scheme seemed simple and obvious enough to me at the time that I thought other people could use it to play a game. Other people apparently never saw it that way.

99. So first you discovered a coordinate system?

Yes, I discovered a coordinate system that was well suited for describing a change in space through time, and it was not logically reducible to an orthogonal system.

100. But you said you had two discoveries. What was your second discovery?

I discovered that life was actually using my coordinate system.

101. That's pretty cool. What did you do next?

I made travel plans to Stockholm

102. Vacation?

I suppose, but I figured I'd need to be there to get my Nobel Prize.

103. Oh. How was it?

Don't know, never went.

104. Why not?

Never got invited?

105. So you never got to Stockholm?

No. But at least I never got Stockholm Syndrome. I never really could empathize with my captors.

106. Didn't you show it to anybody?

Oh yes, I showed it to everybody who'd listen.

107. What did they say, how did they react?

Universal indifference punctuated by intense anger.

108. Why anger?

I don't know, perhaps it had something to do with my personality. I suppose some people don't enjoy being called a complete moron. Or maybe it was the heretical way I attacked their belief system right out of the gate.

109. Well, did you try anything to explain it or convince them?

Yes, I tried everything I could think of to explain it, and then I tried to shame them into at least considering it.

110. Why didn't that work?

I suppose I didn't explain it very well, and they aren't really interested in explanations, as it turns out. To me it is instantly obvious, but that's just me. Plus, these folks don't shame easy.

111. What did you do then?

I gave up… eventually. Screw it; life's too short. I played a lot of golf, invented some cool products to make golf better in the world. I've got a job and a wife and four kids and four cats. Like I say, life's too short, plus it was costing too much money.

112. If you gave up, why am I reading this now?

Well, like I said, I was done with it, but I guess it wasn't done with me.

113. What do you mean?

One day I decided to sell some related toys on Kickstarter, and so I wanted to have a way to explain to people what they were actually buying. After all, it is the first life form on Earth, easily worth $25 with shipping, I should think. Kickstarter advises to "give a description of your project." You might imagine how that went. My description was rapidly turning into another book.

So I started writing a book that might explain it in the simplest possible terms to the simplest possible people. That is the work of fiction to which I previously referred. I thought the story of what happened would somehow be more interesting and entertaining to these folks rather than what it actually is, and then I started having fun with the book and decided maybe I'd sell that on Kickstarter too. That's what I'm working on now. It's my next Kickstarter project, so to speak.

114. So that's what you need my help for?

Yes, that's the premise of the book. If Code World had been discovered and described by a mathematician one hundred years ago and predicted to be the coordinate system used by life, then history would look a lot different. It occurred to me that I might benefit from having a mathematician explain to me what Code World would look like to a mathematician one hundred years ago.

115. What do I get out of it?

I don't know, what do you want?

116. Maybe you are just plain wrong. Have you ever thought of that?

Absolutely. But there is no chance in hell it's wrong. If it is wrong, then I would really appreciate knowing how it can be proven wrong. Even if it is wrong, it's the best wrong thing I can imagine, and it's a lot better than the wrong thing they are teaching poor young school children today.

117. Maybe it's just not useful in any way. Have you ever thought of that?

Absolutely. But that's just silly. It's either right or it's not. As far as being useful, it is incredibly useful if what you are interested in is understanding life. Look at me, look at how well I understand life now, all because I have this thing to help me understand life in the first place.

118. So you understand life better than they do?

I think so. I understand it at the processor level. That's the only way I ever understand anything.

119. Can you explain to me how you understand life differently as a result of this?

Sure.

The first thing you need to realize is what exactly happened to me. I was doing something very specific, and I discovered that life was doing the exact same thing in the exact same way for the exact same reasons. It made sense to me instantly. Why should I ever have to do more than just show it to anybody who also wants to understand it?

120. I can't imagine, but what specifically did you think you were doing?

Building Ralph.

121. You were building a compression algorithm for the universe?

Sort of. It's probably more accurate to say that I was imagining that it might be something we might reverse engineer. I didn't think it would ever be "for the universe." I wanted to imagine a universe where the behaviors are programed into the base and not imposed from the top. So, for instance, think about the sun. What is it? Well, it's a lot of things. It is a sphere. It is mass. It is a source of light. How do you program these things into the base if the base is only a cube? How do you get a mass to be a sphere? How do you get light to travel away from it? When light travels away there are only a handful of truly straight lines, and as these lines get farther away the gaps between them get bigger. How do you make the extra lines, and how do you fill the gaps? I thought that the dodecahedron might provide a viable answer on some basic level.

122. How did that work out?

I don't know. I never did it. I never really got started. At first I was more interested in creating interesting shapes.

123. What kind of shapes?

Fractals.

124. What is a fractal?

Two things. First, a fractal is a shape with self-same geometry. In other words, it is scale invariant. In other other words, it is a shape that looks about the same no matter what scale you look at it. It is a shape made of shapes. Second, from a programming standpoint, it is data that is produced by a function and repeatedly fed back through the same function. I was trying to use the second to produce the first.

125. So, you are pretty good with these fractals?

No. Not really. I just like them, and I think they are pretty cool.

126. How far did you get with this project?

Not very.

127. Does that bother you?

Not really.

128. Can you give me an example of a fractal?

I thought you were a mathematician. You don't know what a fractal is?

129. Can you please humor me?

Okay. The most famous fractal is called the Mandelbrot Set. It is named after the guy whole discovered fractals. It is very pretty. You can look at it on an infinite number of levels of magnification, and it's still pretty.

130. So the universe is a Mandelbrot Set?

I suppose, for lack of a better description. I would think perhaps the universe has more than two dimensions though. I do actually believe that the universe is an information processing system, and the instructions for processing the information are built into the fabric of the universe. I believe that over time the information becomes self-organized, and that through its organization it becomes compressed. I believe that the patterns that this creates in the universe are fractal patterns.

131. And life is contained in this?

Obviously.

132. Let's just stipulate that this doesn't sound stupid. You said that a fractal is generated by a single function, which means that Ralph consists of a single function.

Question?

133. Uh… what is Ralph's function?

Natural selection.

134. And you know this function?

No. Not really. I mean, come on, that's obviously well beyond my skill as a programmer. But I can guess at the things Ralph would actually need to have if he actually had this function. Better yet, I know what the logical consequence of Ralph having this function would be. I know the intent of the function. I know the basic nuts and bolts of how it might work.

135. And those would be?

Well, let's start by giving this function a name, call her Nadia. Let's also give the data a name, call it Bill. Strike that, let's give it two names, Bill and Ted. Now, the nuts and bolts of how it works is this: Ralph introduces Bill to Nadia, they get married, Nadia turns Bill into Ted, and then Ralph turns Ted back into Bill. Obviously, Bill is the universe to Nadia, but of course Nadia would prefer a slightly better universe, and so she turns him into Ted. Equally obvious, Nadia never changes, why should she?

136. Then what happens?

That's it. It just happens again.

137. How long does that go on?

Forever.

138. That sounds pretty boring. Does anything different happen?

Oh yes. Plenty. The whole while this is going on, Bill is getting smaller, and Ralph is getting bigger. That is Ralph's intent, to make himself as big as possible as a way to make Bill as small as possible, at least in the eyes of Nadia.

139. So it's a love triangle?

I suppose, but you'd have to call it a love tetrahedron… if you count Ted.

140. What's the point?

The point is that Ralph is using Nadia to do his job. His job is to make Bill as small as he possibly can. He does this by telling Nadia to pick the things she likes about Bill and then turning him into Ted. Let's call these things she likes patterns. So Nadia tells Ralph the patterns she likes most about Bill, and Ralph makes a note of this and then puts them in Ted. Of course Nadia never runs out of things she likes (or perhaps things she doesn't like) and the things she likes most about Bill are the things that can generate more things she likes. So she likes most the patterns that are able to generate more patterns. The more patterns she picks, the more patterns that are possible. When she starts, there aren't very many possible patterns, but the longer the game is played, the more patterns there are to pick from. Of course this also means that Ralph needs to make more notes. Nadia is very good at not only making choices, but she is really good at making more choices to be made.

141. Where are Ralph's notes?

Well, Bill is the universe, so they must be recorded somewhere in Bill.

142. But Bill is just data, and data is just a line, so what are these patterns and what are these notes?

Excellent question. Ralph started this process by first breaking the data down into patterns that we can call shapes. He then records Nadia's choices in these shapes. So Nadia is basically picking shapes that can do two things: They can record choices, and they can generate more shapes to choose from. Of course this process takes time, but it has to start somewhere. You might say that Bill is nothing more than one gigantic shape, and Nadia is telling Ralph which parts of that shape she likes most. This of course requires a lot of different things, but first it requires a language for telling Ralph which parts of the shape she likes most, and a language for Ralph to make a note of this.

So Ralph needs to do three things: Make shapes, find shapes and record shapes. By far the easiest of the three is making shapes. We don't really need Ralph to make shapes. Recording shapes is hard, but finding shapes is by far the hardest. In this way we can imagine that Ralph's primary goal is a search. Ralph is searching for ways to find shapes that are better at making more shapes so he can proudly present them to Nadia.

143. This is utterly bizarre. Dude, you are tripping.

Yes, a bit like Alice in Wonderland.

144. At least we've finally found one thing we can agree on. Look, this has been fun, but I've got a handball game at 3:00. I still don't understand how this works or what you want me to do. Do you think you might wrap it up and finally explain this to me?

Probably not. It seems to just go on forever, at least in my mind it does.

145. Well, all you are doing is wearing me out. Is there somebody else who might be able to explain this?

I doubt it. I haven't been able to explain it to anybody; that's why I'm talking to you. I was hoping you could help me explain it. The part about the fractal nature of life and the basis of the genetic code is much simpler to see and comprehend. But the only person I know of who seems to understand it is my patent attorney. The only reason he understands it is because he knows nothing about molecular biology, and he's paid to understand how things actually work.

146. Well, did the two of you ever explain it to an expert in molecular biology?

Oh yes. Absolutely.

147. How did that go?

Not well at all. He got very mad at us.

148. What, specifically, did he say?

He said that it was wrong, and experiments had been done “a thousand times” that unequivocally proved it was wrong.

149. Do you disagree with him… do you not believe the experiments?

I do disagree with him. I can’t find the experiments.

150. Did you ask him to show you the experiments?

Yes.

151. Did he?

No.

152. Why not?

I don’t know. I guess he was too mad at me.

153. Well, did you show him Code World?

Yes.

154. What did he say?

He said, “why not a cylinder?”

155. How did you respond?

Well, first I looked at my patent attorney, and we just laughed. What can you say? We didn't know how to respond. So, I said, "Okay, how about a cylinder? Show me one and we'll fight about it."

156. Okay, what about other experts?

I tried. No luck. More of the same.

157. Why are they so hostile toward you? Have you considered that maybe they are all right and the problem is you?

Oh, I know the problem is me. The idea is good; I am bad. I can't even talk to these people. It's not the message; it's the messenger. That's why I'm writing this book.

158. Well, maybe that's the problem: it's not the message or the messenger, but the language of the messenger.

Exactly. That is exactly the problem. That is why I am talking to you. I am trying to change the language and the messenger. The problem is that I have my language and they have theirs, and so there is no possible way to get any message across. The two languages are diametrically opposed. They are mutually exclusive. We are literally yelling at each other through the looking glass.

159. Can you give me a specific example?

Absolutely. I can give you thousands of examples. In fact, I can't read a single paragraph from one of their books without finding a really good example.

160. Well, can you give me just one?

Yes. I'll give you a good one. "Everybody knows" that the genetic code is one dimensional. But what does that mean? Don't hold your breath trying to find out. They won't even define their terms, including genetic code and one-dimensional. What's the point of even saying it? I'll tell you what the point is, the point is that they can then say anything they want, and then pretend they know what they are saying, when in fact they don't even know what they are saying.

161. So you don't think it is one-dimensional?

No. I can't even imagine any universe where it could even be conceived of as one dimensional, let alone ever actually be one dimensional. Because I have a specific universe in mind, and in that universe the genetic code must have so many dimensions of information that I can't even imagine what they all might be.

I can imagine a universe where it might appear, at first glance, to be one dimensional, but it would never actually be one dimensional. In this universe it would start out the opposite of one dimensional and only after a long period of time learn how to fool us into thinking that it might be one dimensional. There obviously is a huge difference between being a certain way and merely appearing to be a certain way. Either you are a certain way or you are not. Big difference.

162. Well, that seems pretty easy. Can they prove that it is one dimensional?

No. But they sure can prove that it is not.

163. Doesn't that bother them?

Apparently not. They're just really happy to have the opportunity to keep on saying it.

164. Okay, how would you say it differently?

There's the problem. It's easy to say it their way and really hard to say it my way. They like their way better, apparently. In order to say it my way, we seem to need to refer to Ralph at some point. My way is the result of an evolutionary process. I think you need to first describe the process before you describe the result.

We know what Ralph is doing and we know why, but how could he ever do it? There are lots of ways. Let's take the simplest. Ralph is trying to make one gigantic shape that he will reference. He is going to do this by finding shapes that will combine to make this one gigantic shape. The easiest way is to find one basic shape and then combine it in every possible way to make the gigantic one. This might work, but it's hard to see how. After all, he has three tasks: Find shapes, build shapes and record shapes.

165. That's a mighty big task to put on that first shape, so what's the next simplest way to do it?

Two shapes. That way Ralph can divide and conquer. Let's just say that is what Ralph did. He found two shapes that he could use to perform the task of building a third larger shape. How does he do this? Let's start by naming the shapes. The two shapes he builds with are A and B, and the shape he is building is C.

We now have three tasks and two shapes to do it: building, finding, recording with A and B.

Well, first, if we are going to do any building or recording, we need a lot of parts. We need a lot of A and a lot of B. So we need a way to make A and B. There are a lot of ways to do this. A can make A and B can make B, and others as well. Let's say A makes B and B makes A. Even for this, we need two things: information and a language, because there can't be information without language. So let's call the language D. Now we have a simple process where A makes B and B makes A using D. What is the information and where is it stored? The information is about making A and B using D, and it is stored in A, B and D, according to how we understand Ralph works.

166. What is the simplest way to do this?

Use a negative.

Think of photography. We take a picture. The "information" of this picture is stored in a negative. We can use the negative to make many prints of a positive. How do we make more negatives? Two ways: we can take more pictures, and we can make more pictures using the positives to print more negatives. By going negative, positive, negative, positive… we amplify the information, but we don't generate any new information. We are interested in doing both. Let's say we take two negatives and produce one positive. This positive will project back to produce a new negative. That simple system will accomplish two different goals in the simplest possible way. We can amplify negatives and create new ones. Still, we need a system of storing this information and a language for transferring it back and forth. A is both a sender and a receiver, and B is both as well.

In the simplest possible terms, this is exactly what life is doing. This is all it is doing. We merely need to translate our understanding of what it is doing into this simple system of understanding what it is doing. A is DNA, B is Protein, and C is all life on earth. D is the language in which it does this.

167. So D is the genetic code?

No, D is a tiny part of what should properly be called the genetic code. You need to understand that A never makes A, and A never makes B. B never makes B, and B never makes A. A and B only uses D to make A, and A and B only uses D to make B. You can never make A or B without A, B and D. Therefore, you can never understand how any of them are ever made without all of them together. So, in order to understand how to make protein, we need to understand DNA, protein, and the language that bridges the two. The information contained in all three is in some sense the same information. The vast majority of that information is contained in the language. There is work done in producing that information, and that work is stored primarily in the language itself.

When somebody says that the language that turns DNA into protein is one dimensional, that is an utterly absurd and demonstrably false statement. In what sense is that true, and in what universe could that ever be true?

168. Well, what do they mean when they say this?

What they mean is this: DNA has four letters. Protein has twenty letters. The genetic code is the relationship between the four letters in DNA and the twenty letters in protein. That is all. Let's call this idea Gomer.

Gomer would be a fabulous theory… if you could prove it. Gomer could vastly simplify our entire understanding of all the life sciences, but Gomer is demonstrably wrong. The simplification it inspires is even more wrong on every conceivable level. They started with a wrong idea and now can't even start in by talking about codons in any useful way, let alone anti-codons, peptide bonds and tRNA.

What they mean now is that it looks for all the world like Gomer is true, and in the absence of any evidence, we prefer to pretend that Gomer is true. The problem is that there isn't an absence of evidence that Gomer is false. There is an absence of evidence that Gomer is true, and an abundance of evidence that Gomer is false.

What I am saying now is that "looking for all the world like being true" is precisely the "goal" of the language in the first place. That's how it ultimately gets its work done. The work of the genetic code is not just building molecules. It's work involves finding, recording and building these molecules. The language itself is taking on more and more of the load so that it can look more and more that way. If we had some way to look at this language a long time ago, it would have no possible way to look anything like this; so it wouldn't. In other words, if you want to understand the origin of the genetic code, the first thing that you need to know about it is that it IS NOT a one-dimensional code. It was even less one-dimensional at its origin.

169. So you are saying we should look at the genetic code how?

Compare the genetic code to the periodic table. For each one, think about what it is, what it represents, what it does, how it works, why we have it, how we got it. Are they the same or different? In what way are they the same or different?

Easiest question: Which one contains more information?

It's not even close. The genetic code not only contains vastly more information, it is vastly more complex – and more powerful. Why would we not intuitively believe that? Because we've been told since birth that it is not true.

At first blush, it would appear that merely as a data table, the periodic table contains more information than the codon table, but even on this score it does not. We normally think of the periodic table as two dimensional, the first dimension representing groups and the other representing periods, but in reality it is a three dimensional table. Different periods have different numbers of groups. Regardless, the table is a data structure that requires 448 bits just to create the structure, and 224 bits to hold the data, a total of 672 bits. The codon table is a three dimensional structure that takes 256 bits to create the structure and 320 bits to hold the data, for a total 704 bits, so the codon table represents a 50% bigger data table than the periodic table. Despite being a relatively trivial comparison, it is the closest the periodic table will ever come to the genetic code by many orders of magnitude.

170. Why do we have them?

We humans have them because they are useful. They predict behaviors. The periodic table predicts the behavior of atoms based on thermodynamic principles. It is predicting likely behaviors of the things we call atoms based on our understanding of how they are made. The codon table is predicting the improbable, and dictating the thermodynamically impossible. There is seemingly no way to use the periodic table to construct the codon table, but that's exactly what happened in the universe.

171. How did we get them?

For lack of a better description, we got them from Ralph. In order to begin to understand this, we need to return to Rex. Let's assume that there is a time zero for the universe. Rex gets in his time machine and uses Liz to capture a chunk of raw data on which Ralph will operate so that Liz can store the data. We need to decide if the universe has things in it or not. Let's say that it does not – no patterns in the data. The first thing that Ralph needs to do is pick a coordinate system to operate on and store the data. At this point, if there are no patterns in the data, there is no such thing as a good or bad coordinate system. In other words, no matter what system he picks, the data will be the same size – it is uncompressible in any system. But when Ralph does pick a coordinate system, it will forever more dictate the nature of patterns Ralph finds in all future data. So he picks one, and off we go.

As time progresses, patterns emerge and Ralph learns them because they are useful toward his goal, which is to compress the data. Ralph is applying Nadia to Bill and forming Ted. Surely one of the first patterns to emerge is the periodic table, because Ralph uses it the same way we do – to predict patterns.

When the periodic table emerges, there is no earth. As a consequence of the periodic table, earth is able to later emerge, so Rex shifts his focus there. In Rex's earliest data captures of earth there is no codon table as we understand it today, but there is a periodic table. The fundamental differences between these two things from this point forward are all a function of time.

The periodic table contains 118 atoms. The codon table contains thousands of atoms. The periodic table contains no molecular structure. The codon table contains mountains of molecular structure.

Think about Ralph working on early earth. He has a table of single atoms, and he has a table for all two-atom combinations, as well as three, four, five and so on. How many of these tables would he go through before he arrived at the codon table? How many are possible? Ralph's ability to pick one table (there are actually many codon tables, not just one) out of so many possible tables represents a staggering amount of information. This information was not given to Ralph, it required work. Information always requires work to create and work to destroy. All of this work is a function of time. Now imagine all the ways that Ralph can use this table to do his job. The table is a pattern, but more importantly, it is the best pattern for creating more patterns. The best way to create new patterns is by finding potential patterns that can create patterns. Ralph loves that.

So, from Ralph's perspective, comparing the periodic table to the codon table, the latter is more task-specific and many orders of magnitude more complex and powerful. It cost Ralph much more in resources to create and maintain, but it pays off in spades in terms of being able to compress the universe. However, this one pattern cannot act in a vacuum. It requires many more supporting patterns in order to do its job.

If you want to argue that the genetic code is in any sense a one-dimensional phenomenon, you better do it now, because this is the best chance you are going to have. It only gets worse the more we look at it. There are so many dimensions of information involved with the genetic code, it is impossible to identify all of them. But the main problem to calling it one-dimensional is an utter failure to recognize the role of time in the code. Time plays a role at every level of its formation and execution. Time alone

would make it at least two-dimensional in any rational sense, but there are many dimensions in any practical sense. A refusal to recognize this is an admission of a severe commitment to total ignorance.

The genetic code at bottom is a compression algorithm. Life uses it to compress and store information. It then uses it to create new information, and to decompress the information it has created. It is important to first recognize these functions, then recognize their proper relationships, then recognize which molecules do this and how it is done.

This description can never exist if we start with "the genetic code is one-dimensional." There is no road to understanding after we take that first nasty step in exactly the wrong direction.

172. How does this put is in a better position?

At least we are finally in a position to better understand life and the genetic code's role within it. Life is an extremely large set of molecules that collectively represent an information system. This system operates on a single, large and complex language. The genetic code is merely a part of this language which at first represents a coordinate system for the information. This coordinate system not only provides the organizational system for information, it provides the logical basis of the language for transposing the information within it as a function of time. This fundamental part of the language and the coordinate system on which it is based then serve to organize all of the information throughout the entire system and all of the language that operates it. The key to understanding life is in understanding the genetic code, what it is, what it does, how it does it and why it works in the first place.

This is not the view presented in textbooks today. This is a view that is much better than the one presented in textbooks today, and I thought it would be immediately obvious, once you show them Code World. It was to me, but I guess I think in different ways for different reasons. It's a more difficult view, but I think it is more intuitive on every level in the end.

The view of the genetic code given to us in textbooks today was given by people who apparently knew nothing about information and evolution, at least not enough to realize that it is all about information and evolution, not just tidy data sets and trips to Stockholm.

After all, building a protein is one thing. Finding new protein to build is another. Building the machinery that builds protein is quite another. The machinery that builds protein also builds DNA. You cannot build one without the other. The machinery builds itself. It is self-building machinery for the purpose of building more self-building machinery.

173. And the periodic table?

Well, now we are in a better position to do an apples-to-apples comparison between the periodic table and the codon table. The periodic table is merely an algorithm for counting from 1 to 118. Conceptually it is simpler for me to see it as counting from 1 to 224, but only 118 of these integers represent atoms. We can then count using base 7 or base 32 and generate the periodic table. Algorithmically, it is simpler still. We can store all of the required information in three bits for every integer in the table. The first bit stores the state of being an atom or not. The second bit stores an end of row flag. The third bit stores an end of column flag. It is redundant to include a row and column flag, but conceptually it is more consistent with the way we traditionally "see" the table when it is printed, and it gives us more flexibility if we actually wanted to generate it. Also, it allows us to perceive every atom as a point on a unit cube. The "meaning" that we derive from these integers is derived from this simple process. The rules for counting give us the structure that is visually so helpful toward understanding the integers they contain. The periodic table literally is a data grid, and it is epistemically correct to use it that way.

174. And the codon table?

Well, how do you count codons? How do you count nucleotides? How do you count amino acids? How do you derive meaning from the method of counting? What is the proper grid for viewing the counting that you have done? The answer is breathtakingly simple. You count with a dodecahedron. With the periodic table we count with a cube. With the codon table we count with a dodecahedron. It not only can be done, but it provides us with a physical structure that is visually helpful toward understanding the integers it contains. Just like the periodic table, it is not the parts that are informative, it is the logical relationship between the parts.

You can say that the periodic table is somehow properly defined by the language of a cube. First we have the language and then we can properly describe the things defined by it. You can also say that the codon table is properly defined by the language of a dodecahedron. First we have the language and then we can properly describe the things defined by it.

175. The task at hand then is to define the language of a dodecahedron. No?

But first, our task is to describe life. My preferred description is that life is molecules doing math with molecules. It is clear to me that the English we use to describe these molecules is hopelessly flawed, but a huge part of that derives from an improper mathematical language. Just as the molecules must have a language to create information, so must we. It is helpful if our language is more consistent with theirs before we start trying to describe them.

176. So this new math of molecules is what kind of math, and how would a mathematician describe it?

We start with regular math, and then we create a new kind of math. We then show that the molecules we know are behaving in a way more consistent with the new math.

The math we know is based on a coordinate system. It is a cube. It is defined by a language populated by integers, call it a counting language. The language of math at bottom is based on the language of counting. We use this language to form a curve which in the obvious special case is a line. A number line is a coordinate system where every coordinate has a unique name and a logical relationship between all other coordinates. Every line is a set whose members are a well-defined sequence of coordinates. There is a perfect correlation between every member and a single location on a single curve.

If we start with a different coordinate system, we get a different language and decidedly different curves. It turns out that counting with a dodecahedron is harder than you might expect. The special case of a line is no longer obvious, and in fact becomes problematic. The best we can do is define a line as a set whose members are a sequence of sets of coordinates. In other words, a curve is a shape made by a sequence of shapes. Each member now has a perfect correlation with a location on that curve, but now there is more than one possible curve, so there is also now a logical correlation between curves. Counting on one curve generates locations on other curves.

177. Why would this be helpful?

This is exactly what life is doing, it would seem.

Life has generated two curves and it is using the correlation between the two to generate more of the curves. In some sense both curves represent a shape, and in a real sense they are the same shape, but one of them is the maximally compressed image of the other. Coordinates from the compressed curve are projected onto the decompressed curve. Coordinates from the decompressed curve are projected back onto the compressed curve, mostly as new coordinates. A line perhaps now can be defined as any correlation between two curves. Life is the zig-zagging of lines between the curves producing a growth in both curves. The same logic that applies to the compression of the curve applies to reproducing the compressed curve. The "math" that can be done is a function of the symmetry contained both in the curves and the logic that correlates them.

I know it can be done, because that's basically what Code World does. I just don't know how to do it. I am not a mathematician. I can get started in an algorithmic sense, but I get bogged down rather quickly. However, if a mathematician were to have done this one hundred years ago and then predicted it was the basis of life, although we might still have trouble understanding it, we'd have no trouble recognizing the value of it. As the actual molecules were discovered they would have clearly been seen to fit perfectly within the system of logic. The counting language of life would be obvious. But, obviously, this is not going to be very intuitive to anybody now. At least it has not been yet.

178. Can you explain it in a way to help me understand it?

Imagine the compressed curve is a single strand sequence of DNA. This literally is a shape. Imagine that the strand contains every possible sequence of DNA. That's a pretty big strand. Now imagine that the decompressed curve is some three dimensional form of the Mandlebrot Set. Every genome is now represented by a single shape on the DNA curve. It's still a pretty big shape, but that shape has a correlation with a section of the Mandlebrot Set, which is a much bigger shape.

Travel from one place to another on both of these curves can be achieved in a variety of ways that take advantage of the symmetry of the curves and the symmetry of the logic that connects them. The most common way is for two shapes on the Mandlebrot Set to project back on a new shape in the DNA curve. This is not sexual reproduction; this is sexual production. Reproduction is when the DNA curve projects on the negative image of itself. The logic that projects from the DNA curve to the negative DNA curve is a subset of the logic that projects from the DNA curve to the Mandlebrot Set. This is clearly

demonstrated by Code World. That's what Code World is. It forms a "counting language" that projects shapes onto other shapes.

We can also easily travel on the curves by merely taking advantage of the symmetry of the language itself. For instance, consider the portion of the DNA curve that projects onto the Mandlebrot Set that we might call insulin. This same DNA curve can be quickly modified by shifting in one direction or the other, by inverting it, or by using its negative image. All of these simple – common – transformations will also represent positions on the Mandlebrot Set, and I'm pretty sure they are "better" than random because of the symmetry of the system. The codon table appears to be well organized by this symmetry in anticipation of this kind of movement.

179. Now you are really starting to confuse me. Can you put into a more concrete example?

Sure, here's another thought experiment that is a little more than mind blowing, but it helps illustrate the main point. Imagine the point in time when you were conceived, call it T_0. Now imagine the time in the distant future when you will die, call it T_1. At T_0 you have a definite location on the DNA curve. Let's just say that is all you were, even though we know you were much more, a whole cell even. Now imagine the projection that is you onto the Mandlebrot Set curve. Now imagine the movement on that curve as a function of time extending from T_0 to T_1. It is staggering. The action develops at such a frantic pace that it is impossible to keep up. But it gets much worse. You are not just moving on that curve, you are moving between the two curves. You can only move on either curve because you can move between the two. That is the whole point, but I doubt that you can yet comprehend the magnitude of the point. The teleology of the whole thing is the movement on the curves.

Think about the basics. You started at T_0 as a single cell. You are entirely defined at this point by five billion digits in the DNA number. You ended at T_1 as seventy-five trillion copies of that number, and all the cells that carry them. How can you have instructions to build something when there aren't nearly enough letters in the instructions to name parts that build it? Those are some pretty efficient instructions, one molecule for every three thousand cells. This doesn't begin to count the cells that used to be you but were left behind years ago. Plus, you have contributed new cells that never existed before. How many grandchildren will you have?

You might only have one gene for hemoglobin, but that gene leads to an awful lot of hemoglobin – trillions of trillions of trillions of it. Compare the gene to its location on the Mandlebrot Set. We won't even go there.

Now lets expand our little thought to include every other life that came into existence on earth the same time you did, T_0. Let's run the tape and see what happens now between T_0 and T_1. All of these things start out as locations on the DNA curve and project onto the Mandlebrot curve. And then there is a jail break, a riot of information processing, the two curves start furiously bouncing back and forth and there's also massive movement within each curve. This is exactly how life is made, and exactly why it causes an exponential increase in information. The name of the game is finding new ways to find new information.

This process scales up and down in time and space. Explain to me how the KEY PIECE of this process, on what scale, in what sense, in what universe could it ever be one dimensional?

I rest my case.

180. Back to the task at hand, how do we count with a dodecahedron and turn this into a formal language?

We start by building an actual coordinate system.

181. How do we do that?

I don't know. I know how I do it with a computer. I don't know how we do that with math, but I know the two are conceptually equivalent.

I'll start with the simplest case. If you want to build a computer program that does anything in space and time, first, you must define space and time. You want to do it in such a way that you get the most bang for your buck. I started in the days of the floppy disk, so storage space was at a premium. What I learned was how to get the most possible information out of a unit of space. In a computer, storage space equals real space.

Coordinate systems for real space, as far as I know, are always based on three dimensional space. There are lots of really good reasons for this, starting with the fact that it is based on real geometry and it is really easy. Still, there is more to it than that, and I'm sure it's not obvious, so I am going to step through the process in painful detail. Then perhaps you can see what I'm saying. We will start with only what we know, decide only what we need, and then move step by step through it. The key to building good data structures is in knowing what you are going to do with them when you are done.

The first thing we need is a reference point. The next thing we need is a name for it. In a computer we only have a choice of two names, 0 and 1. So we will call the reference point 0. Now we need something else, call it a second point, and of course we need a name for that too. Fortunately, we have a spare name. So now we have two points, and their names are 0 and 1. Now we need something else, call it a third point. Now we have a problem – we need more names. Now we need to do something we can't do, so we need to create a system for doing it. What we will do is add existing names together, like a first name and a last name, so call the third point 00. Now we need something else, call it a fourth point. Another problem. What do we call the fourth point? 10? 01? 000?

Anybody who has ever done math, and anybody who has ever written even the simplest computer program thinks I am being stupid, but I am not. This is how you do it when you know you aren't going to have enough disk space BEFORE YOU START. You need to find ways to pack every bit of information into every bit on the disk. There are more ways than you can even imagine. I'm sure some of them will really surprise you.

The only reason it seems stupid at first is because we already know what it means, and we can see that I am doing it all wrong. But do we really know what it means? Am I really doing it wrong? Let me give you an example that will seem bizarre, but I actually do this stuff all the time, and that's why it's so hard for other people to follow my code.

Let's say I have a list of a thousand people, and I want to store information about them. The most obvious information I want to store is their names. I want a first name and a last name, at minimum. Let's say I also want to know whether they are a boy or a girl. There are millions of ways to do this, but let's just assume that space is ridiculously short. Here's the way I do it. First, I create a list that contains every first name and then every last name – one list, not two. Then I store two names for each person. If the first name is first it is a boy and if the last name is first it is a girl. So I don't need to store gender. We can extract information merely from the order it appears. Let's say they also have a pet. I store the names of the pets in the same list with their names. Using the same kind of tricks I can know if they have cats or dogs. And the more I know about them the more I can stuff into their data by creating

clever lists and clever ways of extracting the information. The information I might want doesn't even need to be in any way related to the way it is stored. As long as you know what your data will be and what you are going to do with it, then you can create the patterns that store it most efficiently.

182. Doesn't this sound silly to you?

Yes. This sounds really silly, but try writing a program on a PCjr and you'll learn to appreciate it.

Try putting your own universe on a 1.4 meg floppy and you'll really learn to appreciate it. If you know your data, if you know the nature of your data and the relationships between it, you can really get a cost savings. Of course you have to pay for it with extra programming.

In this case we have a vague understanding of our data – we are trying to build space. So we start with a point, and we need another point, 0 and 1 - one bit. Now we need another point, so we need two bits, but we haven't made three points, we've made four, 00, 01, 10 and 11. Strictly speaking, we have already created a cube. We are given the fact that we are creating space, and we can now find four points in it, so we draw three lines from any one of them and we have the coordinate axes of a cube. What we have not done is define the relationships between any two points. The simplest thing to do is say that every point has the same relationship to every other point, except now what we've done is create a tetrahedron. We can actually do lots of things with this tetrahedron just by the way we store and process the data for it. Still, it's a pretty boring kind of space, so let's add to it.

The conventional way to define coordinate space is with three bits. So now our reference point called 000, and the unique point farthest away from it is 111. In fact we now have eight points. We have two more sets of three points, (100, 010, 001) and (011, 101, 110). What we also have is a butt tonne of extra information. Depending on what we want to do with this cube, we have a lot of different options for storing it. This is perhaps what a mathematician might call data symmetry.

183. What the hell are you talking about?

Well, I'm a programmer. I need to write a program. I want to make things with it. I just made a thing – one thing – a cube. I made it by defining what it is – it is a collection of eight points. I can give each point a unique name, and I can put all eight of them into one structure. So now I can talk about the

structure in specific ways within my program. For instance, I can make a table to store words to describe my structure.

000	Origin
111	Far
001	O_Center
100	O_Right
010	O_Left
110	F_Center
101	F_Right
011	F_Left

This is what one cube would look like if I spelled it out entirely in the computer:

000111001100010110101011

It's the exact same spelling as the table above, but we have trouble reading words in binary. That's why nobody ever does it. It takes three bits to name a point in a cube, so it takes twenty-four bits to name every point in a cube.

184. I still don't understand what the hell you are talking about. You built a cube with points and named the points. Then you spelled the name of the cube with the names of the points. What can you do with that?

I'm building things. In this case I'm building points and building cubes with the points. That's what one cube would look like, but think about how many different ways I could have spelled out this exact same cube. Think about all the things I could do with the exact same data if I wrote a program based on this logical structure of a cube. Think about all the different lines I could get it to draw between the points I already have. Merely by changing the order of the data, I could tell the program to do different things. Think about all the shapes I might create by drawing lines between the points of a single cube. Because I know it is a cube, and I know it has eight points, and I know what those points are and what they "mean" I can write a program capable of doing a whole lot of things with that one measly cube. You'd be surprised. There is an enormous amount of information already contained in the logic of a single cube. A clever programmer could do a lot of things with one cube – if he knew it was a cube.

I will give you a simple example. If I wanted to draw a cube and animate it in time, I could easily think of a dozen different ways to animate it. I could change its color, it's size, its rotation, etc. I would use the position of 000 in the data as my "flag" meaning I want to draw a whole cube rather than just some part of it. I could then use the position of 111 to tell me how to animate it. I would then use the other six

points and their order to specify some kind of color pattern or rotation sequence, etc. So this structure not only carries around with it the names of all its parts, it can carry around instructions that I want to give to the computer to do stuff with it. Similarly, if through the course of time while the program is running I do things to the cube, it can carry around a record of what I've done.

185. Who thinks like that?

Granted, this is not normally how programmers think when they write programs, but it is an option if you need to squeeze in a little "free" information into a lot of the same little things. Sometimes a problem comes down to knowing your options for storing data. This is not how mathematicians normally think. This is not how normal people normally think. But this is one way to think if you are going to try and grow a world from scratch out of a collection of parts. This is perhaps how atoms think when they grow crystals. Who knows.

186. What kinds of things would you want to do with a world like this?

Unfortunately, there are some things I might want to do with a cube that a cube cannot do. For instance, I can make lots of different tetrahedrons, but I cannot make an octahedron. Maybe I can. I can't decide. I guess it depends on how you want to define the rules. Regardless, at this point I can think of two obvious solutions. One of them is dead nuts simple, and the other is a bit harder (pun intended) so let's start with the simple one.

We are writing this program so that we can build things in space. Surely we want to build more things than one cube, but we are building things with cubes, so let's build more cubes. There are lots of ways of going about this, but let's just say we continue on the current path and pick the simplest way. What we'll do is build more points with the same cube. From a programming standpoint, the easiest way to do this is add to the number of names we can give each point. So, instead of using three-bit names, we'll use six-bit names for each point. Now our reference point is called 000000 and the unique point farthest away is called 111111. What we have is a cube that actually contains 64 unique points. That's a pretty luxurious cube. But now also we have exactly one octahedron, and it exists at the six points in the cube called 001010, 111010, 101010, 100010, 101000 and 101011. To somebody as stingy as me, that seems like a pretty steep price to pay for one octahedron.

The other way to do it – a conceptually harder way - is just change the way we see what we are doing. We are creating things and naming things. We know the things we are creating, and we know the logical way they are related and we know the way we are naming them. We know the logical relationship between a cube and an octahedron. A cube has eight points and six planes. An octahedron has six points and eight planes. What if we just switch the things we are calling points and the things we are calling planes?

That seems like a pretty simple fix, but it actually opens a can of whupass on our brains. Does this create any logical difference? I don't think it does, but it has some nasty practical differences. For instance, we like to see that we count points on a line. It's pretty easy to create three lines and start counting. With a cube, what difference does it make if we count with its points or count with its planes? The relationships between points, lines and planes stay pretty much the same. Maybe that's why we like to build things with cubes. The only real difference I can see is that it moves the origin of the cube from one corner to its center. I kinda like that better anyway – it's a more symmetrical way to start building.

But how do we do this with the points of an octahedron? We chose the cube primarily because it perfectly fills space – no gaps. How do we do this with an octahedron since it won't perfectly fill space? How do we address the space not being filled? Doesn't that seem strange? We can count with the planes and the points of a cube in a logically equivalent way, a cube is logically equivalent to an octahedron, but we can only count with the planes of an octahedron. What am I missing? I know I'm missing something, but that one's got me stumped for the moment. I can think of ways around this, from a programmer's standpoint, but I don't like them, and what's the point. Let's just move on.

187. What other things do you want to build with these cubes?

Build more shapes.

188. What kinds of shapes?

Well, let's start with something simple. How about a dodecahedron? We know that all we had to do to make an octahedron was double the size of our cube. How much bigger do we need to make the cube to make a dodecahedron?

189. I give up. Can you just tell me?

The answer is that we need to make it infinitely bigger, and even then we still can't do it. Bummer.

190. What if I want a dodecahedron?

Good damn question. We have two options. We can either accept a close approximation, or we can start with a dodecahedron. Unfortunately, the dodecahedron has the same damn basic problem as the octahedron: Its points won't perfectly fill space. Fortunately, we can solve it the same way we would with the octahedron - start with planes. So that's what we do, we define the planes of a dodecahedron and create a coordinate system that we can use to address things in space through time.

191. Okay, Sherlock, how do we do that?

We go back to square one (pun intended) like we did with the cube and we begin defining and naming things in the exact same way. With the octahedron and the cube it hardly seemed to matter, the choice between points and planes, six versus eight. They are oh so close. With the dodecahedron, no such luck. It has six planes and twenty points. It's a no-brainer. We're gonna start with planes, no doubt. Plus, the planes actually fill space.

192. I thought a dodecahedron had twelve faces. Why do you keep saying six planes?

The twelve faces of a dodecahedron are merely the tops and bottoms of six planes.

There are a butt tonne of ways we might do this, but the consequences are the same. First, we have to "see" our world differently. We have to accept some mighty big trade-offs. On the minus side we are struggling to understand the relationship between point and plane. Add to that the loss of any concept of a line and how that affects our notion of counting. On the plus side, look at the size and nature of our new data structure. It is fabulous. We can do lot's more things with it, including make cubes and

dodecahedrons. But the logical structure really opens up some enticing possibilities. How do we even begin to explore this logical structure?

193. That's when you built Code World?

Exactly.

194. How does that process work?

Well, I did it with shapes and colors. It was much easier for me to see exactly what I was doing, compared to the zeroes and ones of a computer program. But since we don't have shapes and colors here, I step through the zeroes and ones.

The first thing we need are parts, and then we need names for parts. If we pick planes, we need at least six names – it would seem. If we pick points, we need at least twenty names. So let's pick planes. We already know that we need three bits, since two bits only give us four names.

Let's just name the planes by color so we don't get confused by our ordinary sense of direction. Still, we have lots of choices for color, and the way we do it will definitely end up making a difference when we do things in the real world, almost surely. There are only six colors in the entire universe – red, yellow, blue, green, purple and orange. All of the other colors in the universe are combinations of these six. Red, yellow and blue are primary colors – they can't be reduced. Green, purple and orange are secondary colors – they are combinations of two primary colors. So we have a hierarchy of colors. We could rank these according to the "best" colors, like this: Blue, red, purple, green, yellow, orange. But some people might be repelled by that rank subjectivity. Perhaps the simplest way is just a rainbow: Purple, blue, green, yellow, orange, red. This gives us a ring, or a Star of David. Some people might call it a color wheel. The nice thing about this is that it gives us a sense of direction and proportion. It also gives us a single-letter abbreviation for each plane. Now we can make our table for naming planes.

Plane	Color	Abr.	Name
Extra		Btm	000
1	Purple	P	001
2	Blue	B	010
3	Green	G	011
4	Yellow	Y	100
5	Orange	O	101
6	Red	R	110
extra		Top	111

So now we can see that we can build a dodecahedron with its planes, and we get a couple of extra planes in the bargain. The alert reader will notice that this is the same 24 bits as the cube. In other words, were merely need to stop calling it a cube and start calling it a dodecahedron. We will still be able to generate all the parts we need merely by defining the relationships between the parts we already have. So a point is found at the intersection of three planes. This is actually two possible points, a top and a bottom. It's a good thing we have those extra names for stuff like this.

This is what I was doing and this is how I was doing it with Code World. I was speaking dodecahedron to a dodecahedron, or a cube, or an icosahedron, or an octahedron, or a tetrahedron. I was using the vocabularies of a tetrahedron, cube and dodecahedron. I thought this would be the "simplest" way to do it.

What I discovered was that life was doing the same thing, but it had found a simpler way. It is speaking dodecahedron, but it is only using the vocabulary of a tetrahedron. Since it knew it was speaking dodecahedron, and it knew it would be generating points, it picked tetrahedral names for the planes and used permutations of that to generate all of the points. By permuting four names it can generate names for all the points and all of their negatives. In other words, it makes codons and anti-codons simultaneously. It can speak DNA and protein simultaneously. Clever girl.

Hopefully, now you can see what Code World is and what it does, and how useful it is here. It is two things, a coordinate system, and a method for organizing it. What it actually "means" is where people seem to get lost. It makes perfect and immediate sense to me, but that is only because I was doing something very specific when I understood it. Ergo, perhaps you need to be doing exactly what I was doing in order to understand it. You need to be a programmer trying to do a specific thing with a program.

195. And what kind of programmer would do that?

When you are a programmer and you are trying to define the way information moves through space and time, first you need to define space, time and information. There are literally an infinite number of ways to do this, so you need to start by knowing exactly what it is you want to do. From there you need to make choices, and you are going to make trade-offs based on your choices. First you need to choose a coordinate system and then you need to choose a data structure that fits it. Once you choose your coordinate system, everything else will inherit that choice. Once you choose a data structure, everything else will inherit that choice too. You need a "language" that reflects these choices. So if you chose a cube for your coordinate system, from then forward you must speak cube. If you chose an octahedron, you must speak octahedron.

When you pick a data structure, you determine your options for how to use it. As I showed, picking a cube, even with limited data, still presents a lot of choices. But once you start programming with this language you encounter some difficult problems. If you go this way you have some tough problems to solve. I won't go into them, primarily because I was never able to understand them, let alone solve them. At that point, I was happy to entertain any possible option.

Dodecahedron popped up as a potential option. The first thing that caught my eye was that it was not the same as cube. Unfortunately, I could immediately see that compared to cube, it had problems. That was okay, because it offered a lot of potential benefits. The worst problem was the same as the biggest benefit – there are a lot of possible ways to structure the data. Code World seemed like the simplest possible way to organize the data. Once it gets organized, there is a seemingly limitless way to use it, compared to a cube at least.

Once you choose a way to organize it, you also have a language for doing it. This is why I thought Code World was so neat in the first place. It is a game!

So keep in mind what I was doing. I had a lot of stupid ideas. I was wondering about a lot of stupid things, like these:

How do you draw a circle with a cube?

How do you draw a light wave with a cube?

How do you play a fun game with this toy?

More importantly, how do you make money off this toy? How do you get a circle or a light wave to draw itself? What would these things look like? What kinds of rules would you need to put into the data structure of a cube and how would you do it? How much does it cost to mold plastic in China?

The answer to all of these questions apparently is the same: You don't. I thought that you could maybe cheat the circle and the lightwave with the dodecahedron. Maybe the "answer" is in the gaps. I still don't know how to do it, but I do know that I'll never figure any of it out. What's the point?

But I did discover something pretty remarkable: Life is using my coordinate system. It built at least one data structure.

196. Which means?

Life is doing exactly what I was doing for exactly the same reason. What are the odds?

Get it now?

It's pretty damn simple, as far as I'm concerned. All you have to do is see it once, and then you immediately know a lot of things you didn't know before. It will be impossible to convince you that you don't actually know these things.

When we speak atom, we speak cube, and for good reason, it's called the periodic table. When Watson and Crick discovered the double helix, they discovered a data structure. That's why it was such a big deal. They discovered a cube. DNA speaks cube when it makes more DNA with a cube. That's a pretty big deal, I totally agree, but it's not the whole deal. It works perfectly well in as far as it goes, but it only goes so far. They then discovered the codon table and kept on speaking cube. All well and good, but I

discovered something else: When molecules speak genetic code they speak dodecahedron, and when they speak it they get DNA and a language to convert between the two – free of charge.

The vast amount of power toward information processing in this structure is not in the data, it is in the structure. It is in the logical relationships between the pieces of data. In order to understand this, you first need to understand how it can work, and then how it does work. I already understood that part of it, I'd been doing stupid shit like that for as long as I can remember.

I also discovered a stunningly simple way to prove and demonstrate this idea – it's called Code World. It already had a pretty good name. Now all I had to do was find out how much it costs to mold plastic in China, and how many toys I would need to make to pay for my plane tickets to Stockholm.

197. So what's your problem?

The problem that I had, apparently, was that I didn't understand the problem that I had. I thought I was the first person to see something, living in a world where everybody who wanted to see it, and had two eyes to see, could see it too. What I discovered was that I was a one-eyed man with a severe cataract trying to describe something to a world full of blind men with no interest in seeing anything whatsoever.

198. Go figure.

So I've just been busy trying to find a way to describe it. Fool's errand, it would appear.

It's a battle, to be sure, one in which the battle lines are hard to define. That is what I am doing now – trying to define the battle lines. It's hard to win a fight when nobody can even agree what they are fighting about. It appears to me that their position is that I can't do anything, or more importantly, everything I can do they can do too. Nothing short of producing a protein from scratch will do, apparently. I don't have an answer for that, obviously. I can't produce a protein from scratch, but neither can they. At least I have an idea for how it might be done, and I know that I will never be able to do it.

199. So you can actually see how this might be done?

Well, perhaps, but at least I can say I see in a world full of blind men… at least I can see a little bit.

200. So what world are you living in?

I guess it's a world of my imagining. Instead of living in a world in which I never win, I'm imagining a world in which I always win. Of course this world could never exist, and I don't even have the ability to describe a world in which it could exist, but I have the ability to imagine it.

The reason I keep getting my ass kicked is simple. I am living in a world where the fight is being conducted in two languages, the language that they use, and the language that I use. Unfortunately, I am living in a world where there is only one language – their language. It's not hard to imagine why I lose every battle. In my world there are more words than things we might want to describe. In their world there are fewer words than things that must be described. They don't even have words for things they are describing – like molecular information. They always win the argument because they rightly point out that I am not using real words.

201. Can you give me a real example?

I will give you a very simple example. Let's imagine a world where something exists, call it A. Now imagine that two things exist, we could call it A too, or we could call it B. So we have two choices – AA or AB. But by virtue of this simple relationship in this simple world, a third thing also logically exists, call it C. C is nothing more than the logical relationship, whatever it might be, between A and B. C could very well be logically trivial. It does not matter. So now we have a world of ACB. This could be a really simple world, or it could be a complex world. It could be a world that is so simple that A=B, and by virtue of that A=C and so ACB = AAA. At the very least we need words for these things so that we can talk about them. Luckily, we now how words, we called them A, B and C. No biggie. But now let's add an event, call it D. D is a function of time. At T_0 D operates on AAA or ACB, whichever you prefer, and it changes them all to something different, but of the same kind. So now D(AAA) = A'A'A' or D(ACB) = A'C'B'. There is nothing hard to understand about this. This is basic stuff.

Now let's imagine three worlds, call them E, F and G. A person lives in F who wants to talk about his world, but he is unwilling and unable to define his world. He has no workable definition for A, let alone B, C or D, but that doesn't stop him from talking about his world and how it works and what it means. A person also lives in E, and he is a programmer. His only job is to programmatically model the world. He doesn't really care what A, B, C and D actually are, he just needs some words and definition so he can model them. These two people are living in different worlds, and they are speaking entirely different languages, but they need to work together so that they can both be doing the same thing. Now imagine a third person living in G. His job is to coordinate the conversation between E and F. That person is me; I'm the guy living in G. The first job I have is to decide on a common language, and E and F can't even recognize they are speaking a different language. F doesn't even have a single definition of a single word to give to E so he can do his job. How do you think that's gonna end?

202. It sounds really silly.

But this is exactly what really happened. E has everything F needs, but they don't recognize that they are speaking a different language, so nothing ever gets done. All arguments are conducted in F's language, so E never wins, but still, nothing ever gets done. My role in G required that I offer a new language to F, but what use does F have for it? None.

203. Surely you jest.

I'm sure you think I'm kidding, but I am not. We are all actually living in F. In F a thing exists and we call it a codon. We don't have a workable definition for it, but we know we have it. In this world two of these things can exist, but there is no logical difference between any of them so what difference does it make? When two of them do exist, there is a relationship, perhaps a logically insignificant relationship, but a relationship nonetheless, between the two, so what difference does it make? In this world there are plenty of events that will instantly transform all of these things, but they are all the same thing, so what difference does it make? In this world there is no need to define any of these things, let alone give them any names, so what difference does it make? So now what do you want to talk about and how do you want to talk about it? We don't have anything to talk about and it's a good thing because we don't have any words with which to talk about it.

The guy in E might see things a bit differently. Not only does he see a difference, he has a language that could easily handle all of this. The world he models might actually turn out to be so blindingly simple that it is equal to F, but at least he has a way to model it and prove that it does or does not. Until we

have proof, we don't really know what the world is. In fact, he could handle all these issues with a single data structure. I, in the role of G, can't even get E and F to a point where they can talk. It's a language problem.

In E's world he only needs a few simple things to get started. He will identify the things he needs, assign names to them, and then generate the required relationships. Let's call A Albert, B Bernie and C caroline. We now define Albert and Bernie, and see they are different. They might be identical twins, but at the very least they are two different copies of the same thing. We define Caroline as the relationship between Albert and Bernie. Dexter comes along, D, and changes Albert, Bernie and Caroline into Alicia, Carl and Beatrice. All of these things might logically be the same thing, but at least we can see that they are different things. At least we have names for them so that we can talk about them. At least we can finally talk about what they are and how they interact. At least now we can start to talk about what they actually are, and heaven forbid, what they might actually "mean."

In E's world Albert, Bernie, Alicia and Beatrice are all codons. Caroline and Carl are relationships between contiguous codons. Dexter is any change in the reference to a codon. We might name these things something like this: Firston, Secondon, Joinon and Changeon. I don't really like those names, but I don't feel like putting any effort into it just at the moment. What most people don't realize is that there are actually lots of different kinds of codons, and there are lots of different ways to change the reference to them. They think a codon is something that only exists in a cubic spreadsheet. They think the thing in the spreadsheet refers to an actual molecule in life. Unfortunately, there are no molecules that could ever be called a codon; there are only molecules with parts that can be called a codon. And in fact every codon not only carries with it its own specific reference, it also carries its reference to other codons – in real life. The only way to know what an actual codon ever is, is by knowing the specific reference. It's programming 101. How do you discuss these things without words with which to discuss them? I do it by just making up words, but people seem to hate that. They don't know what my words are or why they would ever need them.

The beauty of it, from E's perspective, is that all he needed to do was define a single data structure, and from that he inherited an entire language with which he could have had this conversation.

204. In which world would it be easier and better to live?

Who knows, but that's the world in which I live, a world that is barely battling, but one in which the only battle is entirely over a language. So I am now trying to imagine a different world, one in which the

battle is first and only fought over language before the language could ever begin. It is a language first battle. The language that wins dictates the shape of the world.

205. So what's the problem?

Well, that brings me to the next big problem. I don't know shit about language, let alone how to create one, let alone how to create two and have them compete. As far as I know, all human languages are reducible to the same basic language. That's the only way to get a computer to translate between languages. I can imagine how I might go about reducing a language down to its common features, but I wouldn't know how to actually do it.

I did realize this: no matter what I was doing or how I did it, I would need one thing - a way to count. I would first need a way to count, and then I'd need a way to relate the things I counted. So it seems to me that if I need a different language, I would either need a different way to count or a different way to relate the things I count.

Basically, whatever "new" language I can come up with must come down to a counting language.

Once again, I seem to have a problem. I know how to count, but I don't really know what I'm doing when I count. I know that I am producing things and I know that I am giving these things a unique name, and I know that these things inherit a logical relationship to each other, but beyond that I don't know what I am doing.

So again, I return to Code World. What I think I know is based on a cube. What I don't know is based on Code World. What I hope is that what I don't know is based on a fundamental difference between the two. I don't really know if it is or not, but my intuition is that it is.

When we count with a cube, we don't have a hard time knowing what we are counting. There are a lot of different things that we could be counting, but they all seem to logically be the same thing. Do we count points? Lines? Planes? Or transpositions between these things? What are these things in the first place? They are only what we say they are. So I could say they are anything. I could count them, and then say they are this, or that, or the other. What is the difference between counting the many possible points on the edge of a cube, and transposing the cube on that edge? I don't know. I don't see the

logical differences or understand their meaning. Likewise, I could take a bunch of things that I understand to be parts of a cube, transpose them in any number of ways, count the ways I did it, and then struggle to see the meaning of what I did.

When we look at Code World, we see a different animal entirely. First, what we see is two things, a dodecahedron and a tetrahedron. Now we can set about trying to count all the things in these two things. We can establish the fact that we can count them and that somehow they can all be accounted for by using a cube to count them, I don't know. I don't think you can, but I don't know. But what Code World mechanically does is transpose one thing on the other. In this case, counting things and counting transpositions of things is clearly different. So in this sense it is clearly different from a cube. I don't think you can logically create both things and this system of transpositions if you start with a cube. And stick only with a cube. If you could, I would think it would be extremely clunky.

So, in a basic sense I am saying this. What is a language? It is a system of counting. What is a system of counting? It is a system of naming elements of a cube, which logically reduces to creating a system of performing transpositions of a cube on a cube. What is a new language? It is a new system of performing transpositions of a cube on a cube. How do you create a new language? I don't know.

Try to think of new ways to transpose a cube on another cube. I can't do it.

206. Why would you ever possibly want to?

Well, because now we can actually think of another way. We have A, which is a cube, and we have B, which is another cube, and we have C, which is a transposition of A on B, so A is being acted on in B by C. So, to give you a simple example where we have three different numbers 3, 3^2, 3^3. We could see any of these three numbers in very many ways, but they all represent the same thing. In this case A is the unit cube, B is an infinite orthogonal coordinate system derived from the unit cube, and C is the process of counting to 3, both in terms of unit cubes and axes on our coordinate system. We could translate this in our mind's eye, and in English, into many ways of saying it. We would always start the description with the following: Imagine a unit cube in cubic space with its origin congruent with the origin of space. And then we would proceed with more instructions like, imagine the unit cube transposed in the positive direction three times. Imagine the unit cube transposed in the positive direction nine times. Imagine the unit cube transposed in the positive direction twenty-seven times. You could also do it this way: A Rubik's Cube has three cubes on any edge, nine cubes on any face, and twenty-seven cubes in the entire cube. In each case I am speaking line, square and cube, but they are all based on unit cube.

Now try to think of a way in which you count that isn't the same as this. When we count, what we really refer to is C. We don't actual count things, we count transpositions of things. But the things we transpose are so uniform and well-behaved that we cannot distinguish between the things and their transpositions. Now imagine another way to count. How could you do it? If A is a cube, B is a cube, and C is cubic transformations, I don't see how it is done, so I'm thinking that a new way to count would have to involve making at least one of these things not a cube.

Code World works that way. A is a cube, B is a dodecahedron, and C are transpositions of a cube on a dodecahedron. C is for counting. We count base tetrahedron.

Imagine how bizarre the world would appear. Imagine how bizarre the math would be that operated within it. Imagine how bizarre the language would need to be to describe the math. Imagine how intensely identical that bizarre language would be when compared to bio-molecular structures and their behaviors.

We would first need to abandon our common sense notion of counting. We would then need oodles and oodles of brand new words that describe what we are doing. All of these new words would have very close counterparts already in the world of molecular biology.

207. Can you give me a concrete example?

Sure, go back to the example of 3, 3^2, 3^3, imagine how a description of that sequence would change in fundamental ways. First of all, there is no zero. Or you might say everywhere is zero – no matter where you go, that is zero, as far as the next place is concerned. This sentence: "Imagine a unit cube in cubic space with its origin congruent with the origin of space." Is implied in everything we say when we count normally. That's where you start your counting normally. The new implied sentence is something like this: Imagine the following sequence of base-four units. In other words, you must give the entire sequence before you can reference a part of it. Then the sequence would be this:

231141323443112344321134432

That is the equivalent of a Rubik's Cube, or 3^3, or 27 in this world. But in order to have any visual sense or meaning of this world we would need to perform operations on these numbers too. The first thing we would want to do is project that sequence to some kind of curve in that world, but there are so many

possible projections, and they would all look different and mean a different thing because they would be projections on a different curve of a fundamentally different nature. For instance, we might have an operation that we call project onto the mirror. This would result in the following:

231141323443112344321134432

413323141221334122143312214

This simple function derives from the fact that the transposed unit is base tetrahedron in a cube, and the cube has a mirror tetrahedron, Where 1+3 = 2+4. But we would now also need operations that synchronized other operations between sequences, and they would be called things like Conjoin, Split, Splice, and Shift or invert. We would need operations like Magnify or Amplify. All of these operations would be based on symmetries. Combinations of these operations would sound and look exactly like biomolecules, and they would be based on all the same numerical bases used by the biomolecules themselves. Math in this world would look, sound and feel like life, simply because this is the language of life. Life is counting, and it has a lot of shit to count.

When I built Code World, this is the world I was imagining, and this is the code to which I was referring. That's why I named it Code World. What I realized instantly was it is the exact same code in the exact same world as molecules doing math with molecules. What I saw was the language, because that's what I was making. What everybody else sees is the toy. The toy is merely the foundation of the language - it is the number line, so to speak. I can't get people to stop looking at the toy and start thinking about the language.

208. How would this language change their thinking?

Imagine how the language of Code World corresponds to molecular functions. DNA replication becomes this: Reference Frame, Project Mirror, Split, Project Mirror = Amplify. That might be something analogous to "Multiply by Two." Protein synthesis becomes this: Reference Frame, Project Mirror, Project Dodecahedron, Project Mirror, Project Subset Twenty, Magnify = Solid Base Three. It is a world where sequences generate sequences and operators generate sequences and operators generate operators – they all generate each other within the context of each other. In other words, it's a self-generating world. This is how molecules do math with molecules, especially when all they have available for doing it is themselves.

209. Do you honestly think molecules can do math with molecules?

Well, we are molecules, and we are doing math, so yes, I do think that.

210. Do you honestly expect that to work?

Absolutely.

211. Work for what?

Well, I don't think that it will work for generating a protein from scratch, perhaps, but it will work for generating a language from scratch, and I absolutely think that language will look suspiciously like the genetic code. I think it will create a counting language that brings with it its own math, and that math will generate more of its own language. I think it can be done in a formal sense, at least to a point, and that will get me to where I need to be in terms of writing the story I am writing.

212. But you said earlier that it was the first life form on Earth, and that it provides the best possible clue to the origin of life on Earth. How does that jibe with what you just said?

Everything I have said so far relates to the molecules we know today. These are not the molecules that were around before life was around. Code World gives us the best possible clue as to where they came from.

213. I thought that is what you have been talking about. Weren't you talking about organizing these molecules, simulating them in computer programs and visualizing the language and its effects?

Yes and no. This is where it gets really tricky, but it is also the best place to understand the full value of Code World toward a better understanding of life in general.

Let's start by saying that life is a game that is being played on Earth, and that it has been played from the beginning of Earth. Obviously, this is going to take a great deal of imagination, but we can make quick progress merely by defining some simple terms of this game.

Let us use a game we all know, and then draw simple parallels. Take Monopoly, for instance. It is a game, it has rules, it has a board, pieces, players and a system for keeping score. All of these things go toward the language of Monopoly. Now note that there is a big difference between the game Monopoly, and a game of Monopoly as it has been played. Notice too that although the rules and pieces are fixed from the start, they actually evolve during play. In other words, Board Walk is a different thing at the start when nobody owns it from what it is at the end when it has a hotel.

Starting from that, let's ask what the analogous parts are in the game of Life. First, we have a game called Life, and second we have a game that has already been played called life. I will distinguish the two with the capital letter "L" for the game itself, and the lower case "l" for the game in progress. Next, realize that in Life, the board, all the pieces, all the players, the languages and the system of keeping score are all contained in the original pieces. During play, they create everything else – the game evolves over time.

We can find these pieces in the periodic table of the elements. It doesn't take very long, since for all intents and purposes, they come from the same place – Carbon(6), Nitrogen(7) and Oxygen(8). The only other piece is Hydrogen(1), but it is merely an interchangeable companion for the other three. Together, these four pieces represent about 98% of the pieces in the game, and they will conspire to make everything else in the game – including the board, rules, language and score keeping.

Let's take a quick look at these pieces before we begin. Carbon is a positive ball of six protons surrounded by a negative ball of electrons with four holes. Nitrogen is a positive ball of seven protons surrounded by a negative ball of electrons with three holes. Oxygen is a positive ball of eight protons surrounded by a negative ball of electrons with two holes. Hydrogen is a positive ball of one proton and an electron that fits perfectly into the holes of the other balls.

Let's look at the board. Let's just say that Earth is the board, and it exists in the simplest form. It is a ball with a cover. The ball and its cover are uniform. The game is played within the cover. The cover is nothing but a collection of the pieces frenetically moving around. These pieces quickly conspire to form a coordinate system, or a matrix within which the game can be played; its called water. Water is one Oxygen with its two holes each filled by a hydrogen. It forms a new piece called water that has a

positive side and a negative side. It jostles around with other water pieces and together they form a liquid matrix of moving tetrahedrons. Think of a big box full of tetrahedral dice being constantly shaken. This is water. This is the game board on which Life is played. The other pieces merely find their places within the board.

When I hold Code World, it is hard not to see it as the best possible way for us to see the game board as it would look at the start of the game of life that has been played.

214. That's it?

Oh no, it gets much better from here. That's just the place we can best start to understand how the rest of the game is played. We can ask simple questions about what the point of the game might be, how it is actually played, what are its pieces, how the game evolves and how scores are kept.

The basic point of the game is to have pieces make more pieces with pieces. These, of course, will appear to us as shapes. The method of keeping score is done simply by counting the numbers of pieces of any one kind that exist at any particular time. The shapes that are good at making shapes that are good at making more shapes are the shapes that will be winning at any given time. As the game is played, the rules and languages for playing the game will evolve. They must be stored in the pieces so that the pieces can continue playing the game at these ever-higher levels.

You must understand that this is a fluid game. In other words, there are no static pieces and no static games being played – beyond the simple pieces in the earliest parts of the game. Therefore, it is not surprising that it is being played in a fluid. The pieces don't make eternal pieces that get copied; they make transient pieces that set about making new pieces that are particularly good at combing with and interacting with all of the other pieces.

Now let's look at the starting pieces and try to guess what kinds of higher level pieces they might make. Carbon looks like the best piece on the board. It has four sockets that can be used by four other pieces that are just like itself, which of course is like a bigger version of itself – a tetrahedron. But it also can be used to make chains of itself, or sequences, which are not only good for languages, they are great for keeping score, and great for combining with and interacting with all of the other pieces at all of the increasing levels in the game. Carbon is the natural winner in this game.

215. I thought you said the rules were fixed, but now you are saying that they are evolving. Which is it?

Both. The main rules of the game are embedded in the board and its pieces. Each piece must carry a copy of these rules. But as the game progresses, all of these pieces begin forming more pieces, and each of these must also carry copies of the evolving rules. Together, they define the rules as the game is played. They keep score too!

216. How does Code World help us understand Life at this level?

Well, it is simultaneously a physical model of the rules that Life started out with, and a physical model for the rules that have evolved over billions of years into the pieces that life is playing today. It not only bridges the gap between these wildly different things, it demonstrates the best possible place to start building the languages we might hope to use to describe it. Without this key piece of the puzzle, we end up with languages that just don't make sense, and they end up clouding our understanding of the game and how it is being played. It was a simple mistake with devastating consequences, but fortunately it has a simple fix. I call it Code World.

217. Why not just give them the toy and teach them the game?

I tried that. It didn't work.

218. Don't they like playing games?

Apparently not, at least not if they are biologists and you are playing games with their livelihood.

219. Why not show it to somebody else?

That's what I'm doing. I thought you knew. I already know what they're going to tell me.

220. What's that?

They'll tell me to show the biologist, and come back when they say it's okay. It's like mom says ask your dad, and dad says ask your mom.

221. Why not just create this language yourself?

I can create this language in the sense of a computer language fairly easily, but that doesn't carry much water. What I need to do is imagine a world where this language was created before there were computers and before there was a genetic code. To do that, I think, I need a mathematician. I think it is possible that that would actually happen – maybe it did and we just didn't know about it.

Imagine if a mathematician had created this language and somebody asked him what it was for. What would he say? What is the use for a newborn baby? He might say that it could be used to understand irregular crystal growth. He might speculate that life could use this language to grow the irregular crystals that it needs.

The story I am writing is a hoax. In fact, it is a hoax about a hoax, call it hoax squared. The actual story about the language won't be interesting to anybody, at least it hasn't been so far. The story about the story, I think, will be quite entertaining, at least I plan to make it that way. The thing is, the original hoax is so good, the layman will want the expert to explain why it's wrong. Of course the expert will have no answer for that. He will say, "I don't know what the right answer actually is, but I know it is NOT A." He will be dead nuts certain of his knowledge because he was first taught it in grade school. However, eventually people will start to realize that - hoax or not - a good hoax is much better than no story at all. The search for us at this point is for the best possible hoax. Well, in my opinion, I've already got it. I have the added benefit of actually believing it is true, so I have no shame in promoting it. I don't shame easy.

222. I still have no idea what you want me to do or why you want me to do it or how you think it should be done or why it's a good idea to do it or what it will lead to.

Question?

223. Why do it?

The main reason to do it is to create a literary device to perpetrate a hoax. It is a hoax in every sense of the word. It is not forward engineered for the intended purpose; it is reverse engineered for the stated purpose. The point of the hoax is to illustrate that it is a self-generating hoax. It is a hoax capable of not only sustaining itself, but it is capable of generating all of the necessary hoaxes. It is an illustration of life imitating art and art imitating life.

224. What is life and what is art?

That's the whole point. They generate each other, and you cannot tell which is generating which. In the simplest case, we have me and a literary device. I need the device to perpetrate a hoax. To perpetrate the hoax I need two things: A language and a "time machine" with which to plant the language in my hoax. The device gives me both at the same time. To describe the language I need a time machine, but at the same time, the description of the language generates a time machine.

225. Whoa. I remember you talking about having a time machine, but I don't remember you generating one.

Apparently you skipped that day, or maybe you slept through that class. Ralph is a time machine.

226. No. Ralph is the result of a time machine.

True. But once Ralph is built. He is a time machine.

227. Okay. You definitely lost me there. Can you explain that one?

I'll give it a shot. Ralph is merely a "device" that explains space as a function of time. If you had such a device, you would logically have the essential element of any time machine – a device that describes time as a function of space. You now have the ability to freely convert between time and space. If you are interested in producing new things in space, you specify units of time. If you are interested in producing new things in time you specify units of space. Once you can do one, you can do the other. Ralph generates "disks" that are descriptions of space as a function of time. It wouldn't be hard to imagine a way that translates these same things into disks that are descriptions of time as a function of space. So we can use time to travel in space, or we could use space to travel in time.

So I am an author in search of two devices, when, much to my surprise, I discover that they are two different ways to look at the same device.

228. Okay, I can see how that device might exist, but how would an author deploy that device?

Excellent question. What is author, what is book, and what is device? An author is someone who uses a device to write a book. A device is a way an author creates characters and makes those characters create actions.

229. Your point?

Who creates the device?

230. Once again, you are babbling. Is there something useful in this description?

Yes, I think. We need to create a simple scorecard to see what is happening, because once we let it happen, it's impossible to keep score.

231. That doesn't make any sense. Can you say it in another way so that it might make more sense?

Perhaps. Think of it this way. Think of an Escher drawing. In this case, the best Escher drawing is the one where two hands are drawing each other. We can call this an illustration of an idea, where the idea itself is an infinite egress. Now think of a book as an illustration of an idea. If the idea the book illustrates is this particular illustration, we have now created two things: an illustration of an illustration, and another level of infinite egress. Now we have difficult new layers of infinite egress, which is bad enough, but it gets worse.

232. It can get worse than that?

Oh yes, much much worse. So much worse that you can no longer keep track of how much worse it's gotten. Go back to our original scorecard. We know who the author is, but who are the characters in his book, and what are the actions he forces them to perform? Well, every character is essentially the exact same character: the author. They are either versions of himself, or they are illustrations of real characters that he has met in his real life. Their actions are either those that he created, or those that he has witnessed in his real life.

233. So what?

What if he puts himself in the book? Who are all the other characters and their actions now? What if he also puts characters and actions from real life into the book? Now you have characters creating actions, and actions creating characters, and actions creating actions and characters creating characters. How do you keep track between characters, actions, real life and fictitious creation?

234. You don't. It sounds like an unreadable book. Who would want to read that, and what would they want to learn by reading it?

Aha! Now you get it. It's not about real life or fantasy or characters or action; it is a book about device. What is the device and where did it come from? It's a whodunit, except the answer is unknowable. Obviously, the author dunnit in every book, but in this book who done the device?

235. What the hell are you talking about?

It's a self writing book! More specifically, it is a self-writing book about a self-writing book! Life is a book that wrote itself! It's a book about how you get a book to write a self-writing book! The whole damn book writes itself! The only thing the author needs to decide is how to write a self-writing book, and that's called a device!

236. You are insane. Truly insane. Do you know that?

Absolutely. It's a book about the insanity that has been me over these past many painful years. What's wrong with that?

237. That's just stupid. Nobody will read it, and anybody who does, won't understand it, and anybody who tries will just get pissed off because it can't be understood. Who would want to read that?

Well, anybody who would want to read about the insane world I live in.

238. It sounds like you are Alice living in Wonderland. Is that what you want?

Precisely! I want to describe Wonderland so you can understand the characters living in it. That's the only way you can hope to understand the things they say and the things they do. Except in this case those things are real things and Wonderland is a real place. Every sentence has many meanings depending on the context in which it is placed, and every sentence has many possible contexts. So if somebody says, "it's a hoax," the task is now in placing that in the proper context, for which there are many possible contexts. It's a pretty simple way to write any sentence loaded with meaning. Readers love to read sentences loaded with meaning – those are the best kind of sentences. Wonderland created the real characters in it, and the real characters created Wonderland. Beyond that, understanding the sentence is impossible.

More importantly, the main character is desperately in search of a device which ultimately created him, but he has no way of knowing that. It's called comedy, or tragedy… or both. This results in many

possible versions of him, and he is trying to figure out which one he is. The answer is that he is all of them, which is the obvious case in all things of this nature. He is the author, or a creation of the author.

239. You are twisted beyond all human recognition. You know that don't you?

Yes, thank you.

240. I still don't know what you want me to do. Do you?

Absolutely.

241. Can you explain it to me?

Absolutely not.

242. Don't you see a problem with that?

No. It's not the problem; it's the solution. It's my inability to explain it that is both the problem and the solution. It is the illustration of the problem and the solution together that explains it.

243. Again… gibberish. Care to explain?

Love to. This is a book that wrote itself. This thing you are now reading has drug on so long that we can call it a book. It is a book that describes a book that writes itself. Not to put too fine a point on it, but there are only two characters in this book, and neither one of them are real. Sorry, but you are not real, but then again neither am I, so take comfort in the symmetry of that basic suckyness. One of these unreal characters is merely describing a book that writes itself to the other. This book is itself a book that writes itself. The central action is about creating a device to be used in a book that writes itself. Get it now?

244. I'm not real?

No, but I was really getting to like you. You were really writing yourself with the best of them. Don't worry – I'm not real either. I'm writing myself, and that's okay.

Life is a book that wrote itself. This is a book about life. This is a book that wrote itself about how a book that wrote itself actually wrote itself – it's freaky, in the extreme.

245. My head is spinning. Why are you doing this to me?

I am doing this to illustrate that you are merely a piece in a much larger puzzle, the puzzle of how someone can play a role in creating himself without ever really knowing what he is doing. You are just a character in a self-writing book, albeit a key character… and you are doing wonderfully. Trust me; I've met plenty of characters that couldn't write themselves nearly so well.

246. I'm not feeling so hot, but I can't help but notice that you aren't looking so hot yourself. You do realize that you are saying that you aren't really you, and you are talking to somebody who is also not real. In other words, you do realize that are talking to yourself, and you are demonstrating that you don't understand yourself? In other words, you realize you are totally insane, don't you?

Fully realized, thank you. I am a trained professional in the art of recognizing insanity. It takes a huge amount of insanity to create this world, and more still to keep it going.

247. And that doesn't bother you?

Not in the least. After trying to talk to somebody else for so long – anybody else – I realized that nobody else wants to talk to me, nor do they perceive that I have anything useful to say. I just think there is a fine line between insanity and genius, and I prefer to believe I'm manifesting the latter and not the

former. That's not terribly uncommon, I should think. Maybe not. Who knows. Talking to myself in this way is much better than talking to nobody at all. Call it therapy. Thanks.

248. Glad I could help. What do you want me to do now?

I want you to suck it up and give me that device, buddy boy. Or I should say I want the you that potentially is you to do it. After all, we have already established in embarrassing detail that I am not only you, but also as you I am not able to begin to do it.

249. What?

Do the math.

If we take a quick look at our scorecard, we see that on one hand we have one thing that represents many different things. On the other hand, we have three things that represent one thing. In order to understand this a little better, I will give you some plot details.

There are three central characters in my novel:

The protagonist is a well-intentioned, yet somewhat bumbling 21st century treasure hunter named Bill Parkinson. He stumbles into Code World via a collection of papers that he bought on eBay. The book describes his journey toward trying to figure out what he has and how he can make money from it. The story is first-person narration told through his eyes.

There is a 19th century Hungarian mathematician named Ernö Watrÿck who created the papers that Parkinson later discovers.

There is a Jamaican mad scientist named Harry Stein. He makes all sorts of outrageous claims. He claims that he has been cloning famous scientists for decades. He claims that he can do this from their personal artifacts, and in some cases from their feces. (There's big money in scientific feces.) He claims

that he is a scientist who was born in the 23rd century and was sent back to the 20th century as a young man via a time machine that he constructed. Of course he has been stuck here for decades because, while the machine sent him, it did not send itself, and current technology won't allow him to build another one. His life's work is in trying to build another time machine, and as a way of doing that he has been cloning scientists to help him.

The big reveal at the end of the book – the whodunit – is that these three characters are all different versions of the same character. Parkinson, Watrÿck and Stein are the same guy on different places in the time curve. They are all products of their own device.

That brings us to the device. The device itself is the real central character in the book - not a person. The story is about this main character's evolution throughout the story. It starts life as a language embedded in the fabric of the universe. It becomes a mathematical language invented by Watrÿck. It becomes the genetic code in scientific circles, but they suppress it for selfish reasons. It becomes a toy called Code World. It becomes a self-writing fractal compression algorithm for unimaginable digital storage capacity. It becomes the operating system for a time machine. It becomes the generator of all characters and actions in the entire book. It becomes a self-writing book about a self-writing book.

Then, hopefully, it can do the biggest trick of all – the cherry on the sundae – it becomes performance art. In other words, we create a mathematical language to put into the book. We put the same language out into the real world. The book brings attention to the language and the language brings attention to the book. It is life imitating art and art imitating life – two hands drawing each other. It is what I call my Code World marketing strategy.

250. Sounds to me like you've got it nutted, pal. What's the problem?

From a literary standpoint I have a few problems. I have no problem generating characters because I have so many real characters who have performed real actions that are highly useful to the story. Plus, the characters themselves are creating characters which generate their own actions. The problem I have is that all of this must be described through the eyes, thoughts and actions of a single character – Parkinson the narrator. His abilities are limited by his circumstances. So it becomes a mystery story, one in which each new piece in the mystery must be described by Parkinson. Fortunately, he now has the ability to describe interactions and actions of characters that are living, dead and unborn. That's quite an ability. Unfortunately, his ability to describe the main character – the language – is quite limited. He can only do this by alluding to some of its features and what they mean.

I can do this to some extent, but I finally realized that it would be much easier and better if I had a specific language to which the allusions are being made. This would have the happy paradoxical effect of being able to make the allusions less specific and more effective. People don't want to read a bunch of technical details.

So in creating this language it is important to know exactly how it will be used; that way that it can be used in smaller doses.

251. And the plot is?

The plot so far has Watrÿck working on the problem of irregular crystal growth. His problem is that he can observe this behavior but he has no practical way to talk about it. He knows that it is a process that occurs in space through time, so he needs to have a language or system of shorthand for describing crystal behaviors in space as a function of time. He calls it a language of aperiodic crystals. He speculates that life might be based on such a language. He is a mathematician. He develops his language starting at the bottom – the coordinate system of space and time. He is Hungarian and speaks no other languages. He is virtually unknown and nobody understands or cares about his work. He essentially invents a new system of mathematics, but to everyone else it is all Hungarian. To other mathematicians, it is still all Hungarian. But before he dies, he passes his ideas on to Erwin Schrödinger. Watrÿck becomes completely forgotten.

252. And then?

That's when all the fun begins.

253. The fun?

Schrodinger uses a part of Watrÿck's work to help further his own. He passes it on to a group of scientists that includes James Watson, Francis Crick, George Gamow and Christian Anfinsen. Together they conspire to use a part of the work to further their own, and to suppress the full body of work from public awareness. This is a monumental task, but they effectively achieve it through a secret cabal they

call the Amino Acid Tie Club. In their little club they literally become characters in the story if it were told from another deeper level of the story. This too provides hilarious chapter titles for the book and hilarious dialogue between characters.

Parkinson stumbles upon the work, and he does three things: First, he builds Code World. Second, he investigates the history and meaning of the work and how it was suppressed. Third, he passes it on to Blaine Ray. Ray was a child prodigy in computer compression algorithms. As an adult he was working on the problem of storing, accessing and visualizing the vast amounts of abstract bio-information that science is producing. Ray immediately recognizes the significance of Watrÿck's work toward building a new information compressing scheme, and he sets about doing it. This algorithm, once set in motion, begins compressing data to an absurd degree, thereby creating more of itself to the amazement of all watching. For instance, it compresses Moby Dick down to "Call me Ishmail." It then has the ability to write wildly different variations of the book by merely changing letters and words in that one sentence. "Call me Bruce" writes the story of a Cabaret performer in search of the perfect Judy Garland routine, for instance. Ultimately, Ray has taken the first step in the long journey toward creating Stein and his time machine. And thus the cycle continues.

The rest pretty much writes itself.

254. Okay, this is all very clever and amusing. You have told me what you are doing, how you are doing it and why. Can you tell me what it is you want me to do?

I will give it one more best shot, but you must understand that there is a good reason that it will always all be very confusing.

255. And the reason is?

By nature, it is all very confusing.

What I am saying is that there is a language sewn into the fabric of the universe. This language creates the things in the universe, and it creates languages with and between these things. These things that it creates then create more things and languages. Now there are two distinctly different problems: First, we need to understand these things and languages. Second, we need languages to talk about these

things and languages. It becomes impossible to untangle what is thing and what is language. And it is equally difficult to untangle what is the language being described and what is the language being used to describe it.

256. And that language is?

I understand the concept, but I can't find a language to describe it, at least not one that will allow me to explain it to anyone else. I also can't understand it to a point where I can do anything with it, beyond Code World and this book I'm writing about Code World.

257. Can you give it a shot?

Let's call the language sewn into the fabric of the universe The Law of the Universe. I take it on blind faith that the The Law is responsible for giving us the language of the periodic table of the elements. I also take it on blind faith that the language of the periodic table gave us the codon table. I also take it on blind faith that there must also be a language of the codon table.

I think it is important that we can at least understand this to the point that we realize that the periodic table and the codon table are two different versions of fundamentally the same thing – a projection of The Law. We should at least recognize that they are merely two different versions of graphic representations of things in the universe, things that are using things to build other things. They not only graphically represent the things, they graphically represent the logical relationships between the things. The majority of the information we derive from them comes from the relationships and not the things. At bottom, all of this information can be distilled into a base 2 format of information. In the periodic table it manifests as electron spins. In the codon table it manifests as nucleotide complement pairings.

258. So you are saying that you discovered the law of the universe?

No, I am saying that I discovered a law, a mathematical relationship, and I discovered how this law is capable of generating more of these things that we want to talk about.

259. What is your law?

A dodecahedron is a tetrahedron.

260. That's it?

That's the best I can do.

261. What can this law do?

That's the whole point. Given this law, the rest does itself. There are literally a trillion ways to explain it. I did it in one way, but a 19th century Hungarian mathematician would do it another way entirely. I can explain to you how I did it, but I need you to explain to me how a 19th century Hungarian mathematician would do it. I merely also discovered a third way to do it that had already been done. It's called life. The law not only gives us the codon table, it gives us the parts to put into it, and the language between them.

All three of these different ways wind up looking so similar because they are operating on the same law. Except the 19th century mathematical way to do it will generate a language that is identical in almost every particular to life; whereas, the Code World way only creates visualizations of the law and its parts. I am hoping that this third illustration will finally get people to understand the law and quit looking at the codon table.

262. Okay, can you explain to me one more time how you did it, using the language that you have been using to try to explain it?

Sure.

Give me a dodecahedron and give me a tetrahedron and give me the law and I can give you a language to convert between the two. That is Code World. I looked at it this way: The dodecahedron is the

canvas, the tetrahedron is the brush, and the language is the brush strokes. If I have a dodecahedron and a tetrahedron, I can use the tetrahedron to paint the dodecahedron through a series of brush strokes. All I need to do is break the dodecahedron into its parts – call it the palette – and break the tetrahedron into its parts – call it the brush – and use the palette and brush to define the brush strokes. But I also realized that I could use these same tools to generate all of the other polyhedrons as well. That's why I called it Polyhedrish – a language of polyhedrons by polyhedrons for polyhedrons.

Then I saw that Life had done exactly the same thing, except it did it with molecules. That's why I call it a language of molecules by molecules for molecules. I realized that if a mathematician did this it would be a language of numbers by numbers for numbers. When we spoke the language of numbers we would be doing math. Therefore, life is molecules doing math with molecules.

Simple.

263. Okay, now I get that part. Can you explain to me how a mathematician might actually do this?

Only to a point, and remember, this can't be just any mathematician, it has to be a 19th century Hungarian mathematician. And he has to do it in such a way that it will work best in my book. Even still, there are a trillion ways to do it.

Now you need to understand a fundamental problem. There are two ways to do it; I did it the easy way, but the mathematician must do it the hard way. I used Code World to derive the language. The mathematician must use the language to derive Code World. Still, the process is the same: Given a dodecahedron, a tetrahedron and a law, what is the language?

Here's the rub. A mathematician can't know what a dodecahedron and a tetrahedron are until he defines them. Bummer. The only way he can know what either is, is by comparison to the other, and that basically is The Law. So we must start with four things. First, we should name them. Call them A, B, C, and D, where A is a dodecahedron, B is a tetrahedron, C is The Law and D is the language between them. Now we have many ways to possibly say what we know, and we have many more ways to say any of these possible ways. One way is this: Given ACB we derive D, or perhaps D is implied by ACB. Or you could say that in the context of D, ACB is true. So the mathematician's job is to find a way to say it. I would presume that he would use the language of mathematics already at hand. He will probably use

the words he already knows, and the symbols he already has. I don't know what these are, exactly, but I know generally how they work. One of them is dead nuts simple. The mathematical word for "is" is equals, and its symbol is "=" so we can replace C with =.

Now we can say A=B implies D, or in the context of D A=B. That part is pretty simple, but now we encounter a very ugly problem. We need to define unknown things by their parts. The first thing we need to do is define their parts and then name them. The best language for naming parts is counting. Unfortunately, we can't yet count. That's what we are doing - defining a system for counting. So let's say we just default to the simplest possible system – binary. Now we have a language to build this language, and we have new symbols to do it. We also know what A, B, C and D are supposed to be. We also already know how to describe these parts in binary. D is a counting language base four, which in binary is 2^2. A is a tetrahedron that is a collection of four parts, which in binary is also 2^2. B is a dodecahedron that is a collection of many parts, but in binary the simplest way to describe it – as we saw earlier - is 2^3. C is still equals.

By simple substitution we now have $2^2=2^3$ implies 2^2, or in the context of 2^2 $2^2=2^3$.

The alert reader will notice that we have already made many mistakes. So now we will go in and try to clean them up one by one. One thing we can all agree on is that this language makes no sense to us if we try to use our common sense understanding of what we are doing.

The first mistake we have made is in conflating things with sets of things. We can have four things, we can have a set of four things, and we can have all possible sets of four things. These three things, although all based on four things, are three completely different things. A mathematician already knows this, and he has a language at hand to deal with it.

The first thing that we want to do is say that A, B, C and D are different versions of one thing, call it a set. Now let's call a set of sets a group, and lets call the set of all possible of these sets a symmetry group. There are symbols for these things too, I'm sure, but I don't really know what they are. Let's just say that the symbol for set is this {} and the symbol for symmetry group is this [].

Now we can go back in and try to clean up some of the mess we have made. Let's start with C, or equals. Equals is a relationship between two things, but now it is a set of relationships. It might be a set or a group or a symmetry group, we don't know, but it has to be at least a set. Since the equal sign is so

simple, let's just leave it alone for now, but know that when we use it, we might be using it to mean any one of these three different things.

That was easy.

Now let's look at D, the language. Clearly, here we mean this to be a set of four specific things, so we can write it either as {D} or $\{2^2\}$. Now let's look at A. Clearly, we intend this to mean the set of all possible sets, or all permutations of its members, so we can write it either as [A] or $[2^2]$. Likewise with B we can write it as [B] or $[2^3]$.

There are still more problems to be addressed. First, we need a symbol to mean "in the context of" or "is implied by." I am sure there are symbols for both of these as well, but I don't know what they are. I prefer the former because of my experience in programming. There is a symbol for this, it is this: (). So now we can either write {D}() or we can write $\{2^2\}$(), which both mean "in the context of set D". D is a function, or an algorithm for projecting one set onto another.

The last problem, as I'm sure you have already noticed, is that we are constructing a counting language by using an existing counting language. Clearly, that is cheating. No biggie. If nothing else, we can easily translate our two counting languages into one – binary. Now 2 = 10, and 3 = 11. Obviously, this makes it less clear, but it makes it more consistent.

At least we have cleared up all of the problems – that we know of so far – and we can fully translate the sentence: $\{10^{10}\}([10^{10}]=[10^{11}])$.

264. That "sentence" makes no sense to any possible form of a sentient being. Is that not a problem for you?

No.

At least now we know what we are talking about: We are talking about the translation of a single sentence. No matter how we translate it, that sentence represents a single idea. One way to translate

this idea is this: In the context of a particular language, a tetrahedron is a dodecahedron. Another way to translate the same sentence is this: $\{10^{10}\}([10^{10}]=[10^{11}])$. It is the exact same sentence; you just need to know the translation. Another way to translate it would be this: Give me the right palette and brush, and I'll paint you a pretty dodecahedron. Another way to translate it would be this: Give me a toy and a language and I'll make a game of it. Another way to translate it would be this: Give me some atoms and enough time and I'll build you a codon table. Another way to translate it would be this: Given a bizarre number line I'll give you some pretty bizarre math.

Another way to translate it would be this: This is the first sentence in a book that writes itself. This is a bizarre book about a bizarre world where a book learns to write itself, and then keeps writing more books about it. It is a hellish world that is incomprehensible to both reader and writer. It is some kind of hideous game played only for the sake of playing a hideous game. The only thing that can be understood is that there is such a book, somewhere, somehow.

I know this translation to be true because that is my world and this is one of those books. That is the first sentence in this book, and this book wrote itself. This book is merely the last book in a seemingly endless number of iterations of the same book. I am quite certain that I now have more books than readers. I have written so many books that I can't keep count. I certainly can't remember what I called them, but I do know what I am going to call the next iteration: Code World.

Code World –the novel - has already written itself too, so to speak. It wrote itself by virtue of having written this one, which is an explanation of how to write the next one, and by virtue of the fact that I am forced to live in this world. But the next one will be much more fun, because although I still have to live in this world, I get to make the rules for making the next world. This world in which I live now forces me to play by their rules. In the next world, I get to make the rules and then imagine that everybody must play by my rules. It will end the same way, I'm sure, but it will be fun to imagine living in a world that has to play by a different set of rules, at the very least.

265. Do you honestly think anybody will read any of these books?

Nobody has so far, but I got you to read this one. I'll bet you will enjoy reading the next one too. It will be better, livelier. I would like to think that eventually somebody will want to read some of them.

266. You still haven't told me – in your mind – what is Code World?

That is exactly the problem. I don't know what it is. I need you to help me understand that, because I believe that at bottom it is a mathematical concept. In fact, I am starting to believe that it might actually be a new branch of mathematics. When people ask me what it is, I hesitate to say. When they ask if I can explain it, I admit that I cannot. The problem is that it is so many things, so many different kinds of things, and so many different things within each kind.

267. Can you elaborate a bit on that inherently confusing thought?

Sure. Start with the idea that it is an idea. Code World the idea is the basic idea that the world can contain two logical structures with a simple relationship between them, and this relationship forms the basis of a logical language. From there it becomes a toy, a scientific icon, a computer algorithm - now a novel - and most generally a mathematical concept.

I have always intuitively seen it as a mathematical concept first, but I have never had the ability or motivation to put that into practice. Now that I have that motivation - but still lack that ability - I can see that it is as the mathematical concept of Code World that all of these other things can finally be tied together.

As a mathematical concept, Code World starts with an idea: There are two logical structures whose relationship forms a logical language. This idea must then somehow be refined to a more mathematical statement, like this: In the context of a particular language, a tetrahedron is a dodecahedron. And then this same statement must be translated into a form that can actually generate some kind of math, like this: $\{10^{10}\}([10^{10}]=[10^{11}])$.

This then forms the foundation of a "new math" by first creating a language for counting. That then forms the foundation for counting in space, or what we might call a coordinate system. All of this then forms the foundation for "doing things" with it, what we normally consider to be math; things like add, subtract, multiply and divide, things like drawing lines, circles or cubes, things like plotting data on a curve, things that we ordinarily see as functions of space and time.

It is by wanting to do these things with Code World that we begin to want to understand the fundamental way we ordinarily do them, merely as a way to emulate them with Code World. It is only

then – because of the utterly bizarre nature of Code World the idea – that we finally realize that we have no real understanding of the normal ways we do these things. They cannot be emulated.

Ordinarily, when we do math we start with an intuitive sense of what exactly we are doing. We are using points to form lines which give us numbers. We then easily use these things to form other things, like coordinate systems for adding, subtracting, multiplying and dividing, for doing things like drawing lines, circles and dodecahedrons, for plotting data on a curve, for storing data in an algorithm, for creating functions as a way to do things with space through time. It makes us wonder how we intuitively know that we can do these things in the first place, let alone know how we actually do them and then conceptualize their meaning.

Once we dip our toes in the deep pool that is the mathematical concept of Code World we realize that we must let go of the things that we intuitively feel we must have. But we also realize that we can still do the things we want to do even without them. We recognize that we will need to do them in fundamentally different ways, but this is the best place to start finding fundamentally different ways to do things.

We can no longer use two points to find a line, but we can use two groups of points to find a line, or we can use groups of planes to find a point. We can no longer use groups of points to build a coordinate system that is useful for generating lines, shapes and curves, but we can use groups of shapes to find a coordinate system for generating the growth of both shapes and curves. We can no longer identify points on curves, but we can identify curves on curves or patterns within patterns. It is a bizarre way to try to understand what we are doing, but it is a fundamentally different way of understanding what we are doing that is useful for understanding fundamentally different ways of doing it.

268. How does this translate into the other things that Code World is?

It defines them, extends them, and allows each to build on the other. For instance, Code World the toy is a tangible embodiment of Code World the idea. It can now be seen as a number line. It is a tangible illustration of this number line that provides the foundation of a new kind of mathematics. The mathematics of Code World can be used in its simplest form to play a simple game. The game itself can be seen as a tangible structure with a simple language. However, Code World the toy also serves as the foundation for Code World the scientific icon.

Code World the scientific icon inherits the structure, language and mathematics of Code World the toy, and it extends them. But as an icon it too is many things. It is a physical coordinate system for molecular information. As such it can be used to teach us both the information and the structure of the information. On this score it is every bit the epistemic equal to the periodic table of the elements. However, in fact, it is not one but two structures: The structure of the globe teaches us the logical relationship between nucleotides and amino acids. The structure of the glider (as a cube) teaches us the logical relationship between nucleotides and other nucleotides. But these two structures together teach us a third thing – the logical relationship between the two structures, and this relationship is the logical foundation of the language in which they operate. Perhaps it could ultimately teach us the mathematics of life. At the very least, the mathematics of Code World can emulate the fuller language that life is speaking when it performs its complex brand of mathematics.

Code World the digital algorithm is more bizarre still. It not only inherits these other things, but it extends them into a realm of the truly bizarre. It does this by forcing us to find concrete ways of actually doing it. Once one realizes that one must actually do it, one realizes how hard it would be. We can yet only imagine the results of such an effort. One can imagine the inherent data compression abilities in such a system – the system itself would be based on an idea which is itself a statement of data compression. One can only imagine the iterative power of such a data compression scheme, if it were to actually exist. We might also imagine the nature of simulations that might be performed on such a system. We can imagine curves being plotted against other curves, curves generating new curves. We can imagine the growth of curves. All of these functions would surely have an inherent ability to illustrate fundamental processes of life. What does a protein look like during translation? Probably a lot like the growth of one of these bizarre curves. What does DNA look like during replication? Same answer. What does your DNA look like plotted on the total curve of life? Who knows.

Perhaps the most entertaining form of Code World is Code World the novel, at least it entertains me the most. It inherits all of these other forms of Code World. It is an idea that used ideas to generate more ideas. It is the idea of self-generating ideas. Code World the novel is a laungage of languages that generates languages. It is a self-generating language. It is a book that describes an idiosyncratic journey of a narrator in a quixotic search to understand the meaning of a book that wrote itself. It is a self-generating book. I merely need take the people, places, things and actions from my own life, add a wee bit of artistic license, and voila – a book! The plot of the book merely parallels my journey, which merely parallels the search described here – a search for the mathematical concept of Code World.

269. That sounds like a little more than a wee bit of artistic license, don't you think?

Well, sure. That's why it's a novel. If I didn't take at least a bit of artistic license, what would be the point of making it a novel? Why not just a regular book, like this book, for instance?

270. Okay, I'll bite. Why not this book. You just said that it explains it to the extent you are able. Why not stop here?

Several reasons. As you have rightly pointed out, this book is unreadable. It is unreadable for any number of a large number of reasons. The main reason is that the ideas of this book are extremely hard to understand, so I have essentially distilled out the hard parts and put them in this book. This way, I can write another book that can skip over all the hard parts and merely allude to them "as if" they are real. But the main premise of this book is that Code World starts with the simple idea of Code World. It builds from there and culminates with the realization that Code World, at bottom, must be understood at first as a mathematical concept that defines Code World. This is a fairly accurate portrait of the journey I actually took. This is an illustration of Code World as it is.

This book is written in a form that a computer programmer might explain it to a mathematician. Nobody will read that. It is a piddily little dialogue. I want to write it as a comic opera that anybody might read and be able to enjoy, at least on some level. To do this, I will write it in reverse. In other words, it will start with the premise that Code World started as a fully articulated mathematical concept. This not only generated everything that Code World is; it generated Code World and everything in it. That is the big reveal at the end.

271. Too clever by half, don't you think?

Absolutely, but I think any self-respecting world should run equally well in forward or reverse. If not, how could we ever really understand that world? How could we ever build a time machine? Plus, I think in the end it just represents performance art. Code World the novel is merely a demonstration of Code World the idea. If you like the book, catch the movie, buy the toy, get the T-shirt… that sort of thing. A little of this, a little of that, a little of what have you.

272. WTF?

In other words, Code World the novel is a chicken and egg book. This book you are reading represents both the chicken and the egg. In order to write Code World I had to write this book first. In order to understand Code World the novel you would have to read this book next. It is both primer and spoiler. To fully achieve this effect, at some point the "writer" of Code World must be made to realize that he is merely a "reader" of the book that is writing him. It is a mad cap world, the novel, Code World.

273. Again… lost… direction?

Well, to begin to understand Code World we need to understand it in the purest form – abstraction. Mathematics is the language of abstraction, so we must first understand Code World as mathematics. The mathematics of Code World is the language we must use to ultimately understand Code World. The mathematics of Code World is the mathematics of self-generation. It is the mathematics of growth through self-generation. It is the mathematics of numbers using numbers to make numbers, polyhedrons using polyhedrons to make polyhedrons, curves using curves to make curves, molecules using molecules to make molecules, crystals using crystals to make crystals – life using life to make life.

Code World the novel will have characters using characters to make characters. Fun!

It is true that I cannot even begin to understand this mathematics. It is true that I cannot even begin to make a codon table from a periodic table. It is true that I cannot even begin to make a protein from scratch; but neither can anyone else. They can make a protein, but not from scratch. They borrow the machinery for making proteins, but this does not even begin to explain how to do it from scratch. To do it from scratch one must build the machinery, and unfortunately, that is more protein – and DNA to boot. What I can do is begin to imagine a way that it might be done. I can recognize the epistemic value of trying to start from that point. Perhaps it can't be done, but think of all the things we might learn in the process.

Look at all the things I learned from the process. Before this, I didn't know shit.

274. Well, good luck, Pal. Thanks for letting me trip the light fantastic with you, but I still don't know what you need from me. Should I?

Well, like I said, I'm not a mathematician. I have merely created some decidedly shabby sets and organized decidedly shabby relationships between them. I need a mathematician to define these sets, actually populate them in "the best possible way" and start doing some math with them. Maybe what I have done so far is all wrong. At most it is just a starting point. All I have done is take one simple idea from a larger pool of possible simple ideas. I have picked one possible way from a vast pool of possible ways to position this idea in the starting gate. I have imagined one possible way to give it a start, and look at where it went. Look at all of the things it might actually do. My only hope is that people might learn from this example and realize that this is in fact a valid place to start. Then perhaps they will search for more valid starting places and better ways to set them in motion.

275. And how do you want that done?

Hell if I know – I'm not a mathematician. I thought you knew. I do know that it can be done, and when it finally is, it will serve its purpose quite nicely.

I have called my hypothesis The Language First Hypothesis for several reasons. I feel strongly that the molecules of life must first have a language before they can perform all of the miracles that they have already performed and are currently performing. I believe that this language will be embedded in the fabric of everything they have done and are doing.

I believe that if we want to understand life, we too must first pick a language. Ideally, this language will parallel - to the extent possible – the language of life. It will be embedded in everything we think about and everything we talk about when we think and talk about life. I believe that ultimately for us to do these things we will need to develop many languages with which to do this, so it is imperative that we pick the right first language for all the right reasons. I believe that mathematics is the best first language, and Code World is the best place to begin that language.

276. I still don't see why you need a mathematician to help you do this. What would he do?

Again, I don't consider myself to be a mathematician in any real sense. I like to think of myself as an inventor. As an inventor my priority is to reduce my ideas to practice. An idea only has value if you can reduce it to practice. When I invented Code World I knew that I had invented a number line, I didn't know what that meant. I immediately reduced it to practice, and started practicing. I then reduced it to a simple computer program, and that works too.

In the process, I discovered that molecules had reduced it, and they've had a lot more practice. What I didn't fully recognize was that my number line brings with it a whole new set of mathematics. It is embedded in the number line. It is nothing like anything I've ever seen before, so I can't understand it, let alone practice it.

The thing I now need reduced to practice is the language itself. This will still be a specific language, but it will be of a much more general nature. The two ways that I can reduce it are extremely pragmatic, and they nicely emulate the language of life. I simply did it by creating sets of points and a simple set of their relationships. A more abstract reduction will be much harder and perhaps a bit less pragmatic. It will have at least one nice application, of that I am sure. It will better emulate the language of life and the math it is doing.

I am not a mathematician, but I do know math when I see it, and I can clearly see that these molecules are doing it, and they are doing it with my damn number line. It's going to take a mathematician with far more ability than I'll ever have to even start to understand that.

277. And how do you think this will benefit science?

I am saying that science is not really doing its job anymore. I want to find a way to get science to finally realize that it needs to start doing its job again here.

We have a genetic code today simply because science did its job in the first place. There is a whole lot of great science that went into the genetic code that we have today. Unfortunately, there is also some bad science that went into it as well. This bad science somehow convinced all of science that it could somehow stop doing its job.

With the genetic code we have today, what have we got? We have a collection of parts and a simple system for organizing the parts. That's all well and good, but it is not the job of science to merely give us these things; it is the job of science to explain the things we have. It is to ask basic questions like, what are the parts? Where did we get them? How are they organized? What is the purpose of this organization? How did they get organized in this way to begin with? What is the overall function and "meaning" of all these things we have and like to wonder about?

The answers that science wants to give us today leave a lot to be desired. They care not one wit about the organization, because for them it is all about the parts. This pretty much obviates all other relevant questions. The organization, as far as science is concerned today, is relatively arbitrary. Any one way is basically as good as any other. What difference could it possibly make?

The point of my fiction is to try to get science to imagine what would have happened if the really good science had been done before the bad science had a chance to take root. This would have obviated all the bad science that has been done ever since. The natural starting point is Code World, because it is the least arbitrary way to organize the parts. Also, it gives logical meaning to the organization of the parts, the parts themselves, and the potential origin and meaning of all of these things together. In my book, that's awesome science. But that's just me. The way I imagine this to have happened would require the work of a mathematician; not just a born mathematician but a trained mathematician. This fellow surely could have anticipated all of the good science that was yet to be done, because he would have been working in the realm of pure science, which is the realm of pure abstraction.