Patterns of Intelligence

Chapter 19

Why the Theory of Evolution Cannot Be True

Let us take an example from the "real world" to see how evolution had to work and why evolution cannot be true.

Much of DNA is an incomprehensively complex computer program (e.g. the "morphing of the embryo" algorithms). So let us use a human computer program to explain why evolution cannot be true.

Suppose you wrote a computer program that performed a significant task. For example, suppose it was a very complex "word processor."

The "source code" (which is what computer programmers write) would have been written in a language such as Basic, Cobol or C#, but the "object code" or "compiled code," which the computer can understand, would be entirely '0's and '1's. A "compiler" converts your "source code" into the "compiled code."

We saw a histogram of an actual computer program of '0's and '1's above.

Each '0' or '1' in the compiled code is called a "bit." Let us assume there are 264,000 'bits' in the word processor program you wrote, each being a '0' or '1'.

Note that the computer program (source code in C#) was designed and written by a human being - you.

The computer language (e.g. C#) and interface is a program that was designed and written by other human beings. The computer program source code is typed into a word processor. The computer program source code was written by you and it was written in C#.

The compiler (which converts the source code into executable code) is a program that was designed and written by yet other human beings. The compiler converts something a human can understand (the source code) into something a computer can understand (which is binary or executable code, meaning '0's and '1's).

The operating system of the computer, which executes the compiled code, was written and designed by yet other human beings. The compiled code must be designed for the specific operating system the computer is running.

For example, the compiled code written for a Windows® operating system will not run on a mainframe computer (which is one of the giant computers), because mainframes have totally different operating systems than PC computers.

The hardware microchips were designed and manufactured by yet other human beings who were most likely electrical engineers. The microchip is what actually executes the program.

Other manufacturing companies manufacture the "memory chips" where all of the work is performed in the computer. I have another son who works for Micron, which makes memory chips.

The output of the program may be displayed on a computer monitor. This process is also very complex and every step was designed and executed by human beings.

You can see how many layers of sophisticated "programs" are needed, and they must be coordinated with each other, just to execute a simple computer program!! Each layer or element was designed and executed by human intelligence.

Now suppose that your boss wanted the "word processor" that you wrote to be improved. Let's say he or she wanted 10 more "features" for the program because the new word processor written by your company's competition had a new edition that had these ten features.

Since you were too busy to make the changes, and since your boss believed in the theory of evolution, suppose you suggested to your boss that the new and improved program be written by a random number generator (a "random number generator" generates random numbers in the range which you give it).

Since you believe in the theory of evolution you know this process will work the first time without failure.

So here is the evolutionary process for writing computer programs that you design:

Rule #1) Use a random number generator to choose 25,000 randomly chosen locations on the existing "compiled code" (note that we are changing the "compiled code" that computers understand, not the "source code" that you wrote). The random number generator will choose 25,000 random numbers from 1 to the total size of the compiled code to designate 25,000 random locations on the compiled code. For example, bits #23,987 and #72,108, from your compiled code, might be among the randomly chosen locations.

Rule #2) For each of the 25,000 randomly chosen locations the random number generator would also decide how many additional 'bits', at that location (e.g. from 1 to 10) will be affected by the new changes.

For example, suppose location #23,987 was one of the randomly chosen locations on the compiled code. Suppose for that location the number 5 (which is between 1 and 10) was chosen to be the number of additional 'bits' which are chosen.

Thus, starting at location #23,987, the 'bit' at that location (plus the next consecutive 5 'bits') will be "chosen" for random modification (this is a total of 6 bits).

For each of the thousands of 'bits' that were "chosen" for modification (whether one of the original 25,000 randomly chosen locations or one of the additional bits at each of these locations), the random number generator would also decide whether:

a) The bit was deleted from the compiled code, or
b) A new bit would be inserted right after that 'bit' (and the 'bit' that was inserted, a '0' or '1', would be randomly chosen by the random number generator), or
c) The existing bit was inverted (i.e. a '0' was changed into a '1' or vice versa),

For example, in slot #23,987 there might be a '0' in the original program. Suppose the random number generator selected "inverted" for this slot. The program will replace the '0' with a '1'.

The program will then physically make the changes to the compiled code and save the new word processor program, including each of the bits at the 25,000 different locations plus the additional bits.

After doing this, our "new" word processor now has had many thousands of totally random "mutations" or "changes" to the compiled code of your original compiled program.

Here is the key question: will the "new" compiled code be a vast improvement over the "old" compiled code and will it have the ten desired changes your boss wanted?

Well, if both you and your boss believed in the theory of evolution, both of you would agree the new program would have the ten new features and would work even better than the old version.

But the real answer, which would be obvious to any computer programmer, is 'no'. There is not a snowballs chance on the surface of the sun that the "new"

compiled code would be an improvement over the old code; nor would it include any of the new features!!

In other words, there is a 100% chance that the program had zero new features and a near 100% chance the program would not even run (i.e. execute).

But the key point is this: the new program would ***NOT*** be an improvement over the old program!! There would be no new features, not even new features that you had not planned!!

Why, you might ask?

Let me explain. Suppose you were to intelligently write the new computer program your boss wanted. Suppose it had all 10 new features.

What are the chances your new compiled code, and the compiled code created by the random number generator, were the same?

The answer is zero. There is no chance the two compiled codes would even remotely be similar.

Let us look at three of the reasons why.

First, is the "location" issue, meaning where will the mutations occur? If you could see where a random number generator selected 10,000 locations on a DNA strand or computer program, you would see that the locations of these mutations will be *fairly evenly spread out* over the entire length of the DNA strand or computer program. That is the nature of randomness.

However, when making sophisticated changes to a computer program, by a human being, the locations of the changes, which are intelligently made, would be *largely clustered* in certain locations.

For example, consider the changes to the compiled code for one of the new features. This section of code would include a large number of changes to at least one small section of the compiled code (where the main section of the new algorithm was located).

But randomness does not cluster changes, it spreads them out evenly.

Second, is the concept of "permutations." Consider this section of new code which you wrote to satisfy your boss (spaces are added to make it easier to read):

```
00110011 00111100 10101001 11111001 01000001
```

Suppose this was a permutation (i.e. a unique ordering of 40 'bits') that **must be** in the new computer program which was not originally in the old program.

What are the odds a random number generator would come up with this exact permutation of 40 bits if it generated 40 random bits? The odds are 1 in 1,099,511,627,776. **That is less than 1 in a trillion!!**

But even this does not take into account that these 40 bits are clustered together and that randomness does not cluster things.

It would take many, many, many billions of attempts of writing a new computer program before you would see this permutation, depending on how large the computer programs were, even if you intentionally clustered these 40 bits!!

Now consider: what if 10 different required permutations, of this length of 40 bits, were required in the same computer program? You would never see 10 such precise permutations in the same computer program if you tried 1,000,000 times a second, 24 hours a day, for a trillion trillion trillion years.

Now consider that the complexity and sophistication of human DNA is far, far greater than any computer program ever written by a human being.

Much of human DNA is like a complex computer program. Requiring 20 consecutive nucleotides (which is the statistical equivalent of 40 computer 'bits') to be exact nucleotides would be a common requirement and could be required several times to create an advanced new species (a "child species") from a prior advanced old species (a "parent species").

For example, genes are sections of DNA code which require very precise sequences of codes. And this code is largely clustered together. Genes are what make proteins. Proteins must fit together very precisely when they are used to create a "protein structure." There is very little, if any, margin for error.

But randomness could never create such sophisticated changes to DNA, nor would the changes be clustered.

Third, there is intelligence in computer programs when they are written by human beings. We saw a very small example of graphically visualizing intelligence above, but it is obvious that intelligence is not going to be generated by a random number generator. We also saw that in the above bar charts.

Try writing a new computer program, from scratch, with a random number generator, and then see if your new computer program does something intelligent? It will never happen.

The Point

Evolution claims that zero intelligence directed evolution. This means that changes to the DNA of a "parent species" had to be randomly made to create a "child species." The "locations" of the mutations (i.e. where on the DNA is the change) and what was done at those "locations" had to be totally random.

Also, with evolution, any "child species" that survives is always assumed to be a superior species to its "parent species," which is nonsense when you assume the changes to the DNA of the "parent species" were randomly done!! It is impossible the "child species" could even survive, much less be superior to its "parent species."

But if randomly modifying a computer program, which is far simpler than DNA, cannot create a superior computer program; and if DNA is many times more complex and sophisticated than any computer program; how could randomness create a new and improved species?! It can't. And that is the point!!

To say that random mutations to DNA (including random locations and random changes) could create a new species is far beyond ludicrous. And to say it has happened millions of times on this planet, in a mere few million years, **with few or no errors**, is absurdity beyond comprehension.

Actually, in your personal life you already know that the theory of evolution is nonsense.

For example, suppose someone at work walked up to you and handed you a CD with a new version of a software program your company makes.

You would immediately think that highly trained computer programmers in your company made the updated version of the program.

You certainly would NOT think that someone took the prior version of the old source code or old compiled code and applied a random number generator to the old source code or old executable code to make the changes!!!

As another example, if the programmer claimed the new version of the software was "programmed" by a random number generator; you would give them the "glare" of sarcasm to make it clear you knew the person was playing a practical joke on you.

But evolutionists think that the millions of unique species on this planet (including the "first living cell') all came to exist essentially by a random number generator (i.e. "evolution") randomly mixing up the DNA of a new or previously existing species!!

What nonsense!!

More will be said about these issues later in this book, but for now I wanted to give you a quick overview of how "macroevolution" had to work.

Hopefully, the reader is beginning to understand why evolutionists need to deceive their students by using examples of microevolution as "proof" of the theory of evolution.

Never, in the history of this planet, or any other planet, has macroevolution created a new species.

I am not saying that it is statistically impossible for randomness to create a complex new gene, what I am saying it that it is so mathematically absurd that it would not likely happen more than 2 or 3 times in the age of our earth, given the speed of biology (as opposed to the speed of computers). And these 2 or 3 times would definitely have been on single-celled species which had very simple DNA (i.e. short genes).

Macroevolution is not statistically impossible, it is only statistically impossible in the sense of saying that millions of unique species, many of them with very long and complex DNA, were created on the same planet in the space of a few hundred million years or even a few trillion years. Evolution works a lot slower than computers. This is just one reason that evolution is statistically absurd.

Someone might say that the probability of evolution is like picking the correct single atom from among all the atoms in our Universe. No, that is not correct. The probability of even simple evolution being true (i.e. going from a mouse to a slightly more complex species) is more like picking the correct single atom from among 1,000,000,000,000 Universes. Actually, it is far, far worse than that especially if it must be done during the age of a single planet.

How would you like to sit in a chair and stare at the moon until a lizard was walking around on the moon via evolution (meaning it was not put there by astronauts or space ships)? You would be there forever even though there is a lot of water on the moon and even if someone put plenty of air on the moon.

My point is that it is critical to understand these terms and watch for evolutionists to use examples from microevolution as "proof" of the theory of evolution because there are zero examples in this Universe of macroevolution creating a new complex species (which, in many cases, would require the random creation of several new and complex genes and many other things), much less the creation of millions of new species!!