The Blueprint for Life?

Dror G. Feitelson
School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel

Millet Treinin
Department of Physiology
Hebrew University Hadassah Medical School
91172 Jerusalem, Israel

Abstract

The genome projects being completed today, and especially the Human Genome Project, have created a great interest in understanding the information encoded in DNA. In this context, DNA is often referred to as "the blueprint for life" — implying the assumption that all the information needed to create life is encoded into the DNA. But this approach ignores the complex interactions between the DNA and its cellular environment, interactions that regulate and control the different spatial and temporal patterns in which DNA is expressed. Moreover, the particulars of many cellular structures seem not to be encoded in the DNA, and they are never created from scratch; rather, each cell must inherit templates for these structures from its parent cell. Thus it is not clear that all life processes are directly or indirectly encoded in the DNA, casting a measure of doubt on the concept that they can be understood solely based on the study of its sequence.

Introduction

Bioinformatics, genomics, proteomics — these fields of study have exploded into the public eye in recent years, with the dramatic mechanization of DNA sequencing at an industrial scale, and with the advent of microarrays that allow the expression patterns of thousands of genes to be quantified simultaneously [4]. These breakthroughs are already beginning to deliver in terms of new ways and means to classify tumors, and new insights on how to fight viral infections. Such achievements are based on better observations on one hand, and better understanding of cellular processes on the other. This has also opened the door to more basic questions involving the interplay of different components in a cell's life cycle. In particular, we are interested in the degree to which "everything" is encoded in the DNA.

One of the greatest scientific discoveries of the twentieth century is the structure of DNA and how it encodes proteins. The simplistic view, taught in high school, is as follows. DNA contains the information needed to create proteins, namely the sequence of amino acids that needs to be stringed together. This information is transcribed into mRNA, and then translated by a set of mediators (which are also encoded by DNA) including ribosomes and various types of tRNA. Once created, the proteins interact with each other in the cellular environment, leading to the biochemical processes of life.

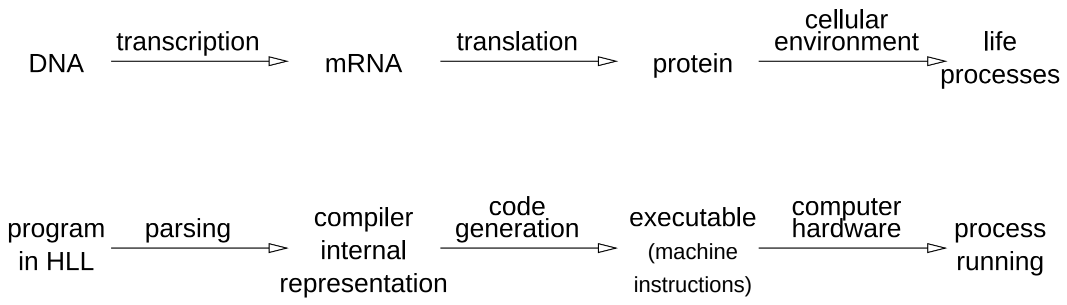


Figure 1: *The simple model: a linear progression from information to function.*

An analogy to this process can be found in the workings of computer programs. Programs are originally coded in some high-level language (HLL), which specifies the sequence of instructions that is to be carried out. This is translated from the abstract high-level language to an executable form by a compiler (typically in two steps: parsing and code generation). Finally, the executable is brought to life by its interaction with the computer hardware during execution.

The basic feature of these descriptions is the simple linear progression from information on how to create something, to the creation of that something, which is then animated and functions in its environment (Fig. 1). In particular, it all starts with the information, and the information encodes all that is needed to proceed. The translation is merely a mechanical process of deterministic substitution of one symbol for another.

The case of DNA is special because the program it encodes — if we choose to regard it as a program — is for the reproduction of its environment, including itself. In computer terms, this would mean that the program's function is to build new hardware and run on it as well. Obviously, this requires raw materials to be acquired and assimilated, in order to fashion new structures. But what about tools? While one may claim that the instructions for creating the required tools are included in the program, one must also agree that some initial toolset must be present in order to set the process in motion. Such tools, in effect, convey information about the function of the system as a whole, information that is not necessarily available in the program. Moreover, switching the available tools may modify the outcome, thus diverging from the construction of a true replica. In the following sections we explore these ideas in a biological setting, in an attempt to characterize those pieces of information that are required for life but are not encoded in the DNA.

DNA: A Medium for Information Storage

The DNA molecule is an amazing example of nature pre-staging humans. It is not the only one. For example, birds and even insects could fly before humans could, and bats' use of the reflections of ultrasonic sounds is often referred to as pre-staging the development of radar and sonar systems. But DNA is different. DNA is not like any mechanism that operates on objects in the physical world. DNA is as close as one can imagine to the concept of pure information storage. Moreover, it uses a digital encoding.

When the structure and function of DNA was deciphered in the 1950s, it turned out to be remarkably similar to the abstract tapes postulated for storage in Turing machines some twenty years earlier, and the magnetic tapes used for storage in early electronic computers. DNA, like a tape, stores data in a linear sequence. The data comes in discrete units, which can be regarded as letters from an alphabet. In the case of DNA, the letters are the four types of nucleic acids, commonly denoted by A, T, C, and G. These letters are then grouped into words that are three letters long, in order to encode a larger set of symbols (the amino acids described next). In electronic computers, the basic alphabet is binary (zero and one), and words of 8 or more such basic symbols are used to encode letters in human languages and numbers.

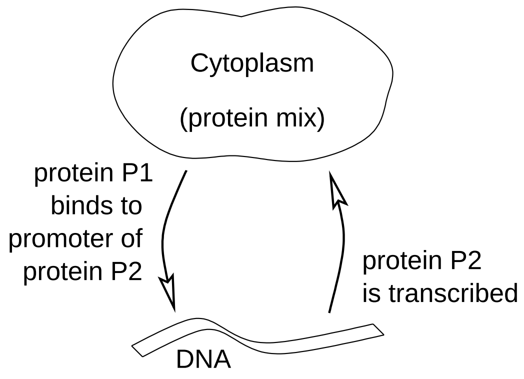
The most important information encoded in the DNA is the instructions for the creation of proteins. Proteins, like DNA, are molecules that are essentially a linear sequence of building blocks that come from a very restricted repertoire. In the case of proteins, these are the 20 amino acids. But proteins are not used to store information. Rather, they assume the roles of both the infrastructure of living cells and the agents that "make things happen".

The process by which the information stored in the DNA leads to the creation of a new protein is rather involved. First, the DNA is transcribed into messenger RNA — essentially a copy that can be sent to the production plant. This is complicated by the fact that a gene need not be stored in a contiguous stretch of DNA, and the non-coding parts (the introns) have to be spliced out, leaving only the coding parts (the exons). The production plant — embodied by ribosomes and tRNA — translates the mRNA codes into amino acids, and facilitates their linking into a protein chain. All these steps are catalyzed by existing proteins that bind to the DNA and the RNA. The ribosomes themselves, which provide the infrastructure for this process, are also largely composed of proteins and RNA. How all this came to be is one of the major questions facing science today [13].

It should be noted that the information contained in the DNA provides only a partial specification of how proteins should be created. The missing part is the three-dimensional structure that should be created from the string of amino acids. The difference is the solution of the protein folding problem. It may be that once a protein is created, it immediately folds into the correct configuration, and this configuration is unique. But it seems that in most cases the same sequence of amino acids has more than one stable configuration, and the one that takes effect depends on other proteins (called chaperones) present in the environment in which the translation takes place. If the appropriate chaperones are not present, the correct functional form of the protein will not be created.

The Cell as a State Machine

In nature, DNA does not have an existence of its own. Even viruses, the nearest thing to "pure DNA", need a cellular environment in order to function. By themselves, they are inanimate storage objects. The special thing about viruses is that they are constructed so that when placed in a cellular environment, proteins present in the cell operate on the virus DNA, and incorporate it onto the cell's reproductive cycle. In a sense, this is similar to what was done in recent mammalian cloning success stories: the researchers put DNA from one particular animal into an appropriate embryonic cellular environment from another animal of the same species, such that this environment took up the DNA and entered the normal


Cytoplasm
(protein mix)
protein P1 binds to promoter of protein P2
protein P2 is transcribed
DNA

STATE = {set of proteins}

DNA rule:
if P1 ∈ STATE then
  STATE = STATE ∪ {P2}

Figure 2: *Simplistic view of the cell as a state machine: the expressed proteins are the state, and their interaction with the DNA cause new ones to be expressed, thus changing the state.*

differentiation and growth process [14]. This was done without a full understanding of what the environment contains and how this process works.

The fact that proteins bind to the DNA in order to activate the transcription process indicates that the DNA contains additional important features, not only the explicit data about protein sequences. The most important binding sites are the promoters, which activate the transcription, and splicing sites, that allow the sequence to be constructed from several disjoint pieces. In addition there are sites that cause transcription to terminate. Collectively, all these sites can be considered as control features that augment the data itself. The control sites link the DNA to its environment, by allowing the environment to influence what proteins will be fabricated under what conditions. Cells generally do not create most of the proteins that are encoded into their DNA — only those that should be, as dictated by other proteins that interact with the DNA. This is the main difference between the genotype, i.e. the full list of genes, and the phenotype, i.e. those whose effect we see.

In computer science terms, a gene can be viewed as a state machine's transition rule. The current state of the cell is determined by the composition of the cytoplasm, that is, by the mix of proteins that exist in the cell. Some of these proteins may interact with the gene's promoter in order to activate it. This eventually leads to the production of a new protein, whose addition changes the state of the cell (the mix of proteins; see Fig. 2). The genome is a collection of very many of these rules. From any given state, there are transitions to many other states according to whatever rules are enabled at the moment, meaning that their gene's promoters can become active. In a real cell, many of these transitions may take place concurrently, as opposed to the sequential nature of the state-machine model.

The use of embryonic cells for cloning is an example of the notion of states: for cloning to work, the desired DNA has to be injected into a cell that is in the "initial state" of the machine. Only such cells have the correct mix of proteins to start the differentiation process. Naturally, the cellular state machine is very complex. The fields of gene expression profilling and proteomics are devoted to decyphering this state machine. An important tool in this quest is the microarrays that enable the expression levels of thousands of genes to be recorded and quantified simultaneously. This essentially ammounts to recording the cell's state. As the state correlates with various normal and pathological conditions, being able to identify the state is equivalent to diagnosing the respective conditions.

It should be noted that the above description is very simpleminded, and ignores many of the real complexities known to Biologists. One of the complications is that many transitions occur at once. In addition, the feedback loop from the proteins to the control of gene expression is actually more complex: feedback occurs at all stages of protein production, not only at the initial transcription [4]. Thus existing proteins affect the splicing of mRNA, its transport to the ribosomes, and the folding, structuring, and stability of the newly created proteins. All of these stages are loci for the exercise of control. The complexity of these processes, best examplified by the process of transcription, implies that it is hard to assign full control to the information encoded in the DNA. For example, transcription may be regulated by multi-protein complexes whose formation depends in a non-linear manner on delicate quantitative balances that integrate inputs from many different signalling pathways [7]. Nevertheless, we will regard the transition rules as emergent properties of the relationship between the DNA binding sites and the complex states of the cell.

While the full state machine is very complex, it is actually composed of a large number of largely independent smaller state machines. Skin cells, by virtue of being skin cells, occupy a different part of the state space than muscle cells or nerve cells. And the different types of cells retain their identity across cell divisions. However, the different state machines are coupled to each other after all. This is obvious from the process of differentiation, where a whole organism starts from a single cell that divides repeteadly. Each daughter cell assumes a different role — moving to a different area of the state space — based on external stimuli from its neighbors. This seems to be a uni-directional process: once the cells differentiate, they indeed go their separate ways.

With this veiw in mind, we can observe that a living cell involves the interaction of *two types of information*: the transition rules largely encoded in the DNA, and the state encoded in the cytoplasm. Neither alone can suffice. And neither is more important than the other.

Non-DNA Information

As we established above, the information about a cell's current state is embodied in the composition of the cytoplasm. But can the cytoplasm affect the developmental trajectory in a way that is independent of the DNA? In other words, can changes in the cytoplasm be inherited, without corresponding changes in the DNA?

It would seem that the role of the cytoplasm is only to interpret the DNA, by providing the enzymes and structures needed for the translation and transcription. This is much like the process of compiling a program to an executable. But in computer science, we know that dishonest compilation is possible, resulting in an executable that does not exactly reflect the source code (see sidebar on Trojan horses). Similar effects can also occur in living cells. This is called epigenetic inheritance, because inherited traits are not passed through the conventional genetic mechanism. Rather, they are passed as functional biochemicals (typically proteins) that exist in the cytoplasm and mediate various effects. As cells split, their cytoplasm passes to the next generation, together with these proteins [6]. Remarkably, this happens even in unicellular organisms such as bacteria.

Although epigenetic inheritance is not the common case, several examples have been found. Many of these depend on modification of DNA structure without changes in the code itself (e.g. methylation and chromatin remodeling) [9, 6, 5]. As such modifications may affect expression patterns, it is hard to claim that they are independent of the DNA.

However, examples that are not related to the DNA at all are also known. Perhaps the best known example comes from prions, infamously known for their role in mad cow disease. Prions are proteins that have two different stable conformations, that have different functions. The crux of the effect is that one of the functions is to control the conformation of similar proteins. Thus once a protein adopts a specific conformation, it causes other copies of itself to do likewise. As the proteins propagate in the cytoplasm from each cell to its daughter cells, this preference propogates too [11, 3].

Another class of DNA-less inheritance is the inheritance of structures. Perhaps the best examples come from *Tetrahymeana* and *Paramecium*, two protozoa that have cilia protruding from the surface of their cells. These cilia are organized in complex patterns, and are capable of rhythmic motion that is used in feeding. An extensive analysis showed inheritance of variations in these patterns that is independent of DNA sequence [8]. Moreover, experimental manipulation of these structures also led to inheritable changes [1].

While known examples of inheriting structural modifications are rare, the actual inheritance of structures is very basic. There are many cellular organelles that are required for life but are not created directly from DNA-borne specifications. These organelles are typically replicated independently — they grow and split independently of the encompassing cell cycle (albeit at about the same rate). For example, membranes which are crucial for cellular functions are composed of lipids, not proteins. The source for lipids is external — they are an important part of food intake. They are then modified as needed (by proteins), and assimilated into existing membranes. As membranes grow, they allow the cell to grow and eventually to split. This is not directly encoded in DNA.

Similar processes occur in other organelles. Mitochondria are not created from scratch — they grow and split [15]. The endoplasmatic reticulum is built based on existing structures that are extended, and not recreated from scratch. The same applies to special parts of it, such as the the Golgi apparatus [10]. In all these examples the raw materials are indeed proteins, but they do not naturally come together to create the required complexes. The organelles are composed of many different types of proteins, which must connect to each other and be embedded in lipid membranes in the right ways. Thus the organelles have to be there to begin with, in order to assimilate the new proteins into the correct locations of the structure. By doing so they grow, and eventually they can split.

Summing up all the above leads to the following five-tier model of information needed for cellular life:

1. The classical well known encoding of proteins in the DNA

2. The DNA control regions, by which existing proteins control the transcription of others

3. Indirect control by interactions among proteins, e.g. the creation of complexes and support for trafficking and assembly

4. Propagation of the contents of the cytoplasm from a cell to its daughter cells, consequently affecting their behavior

5. Structural information, embodied in cellular organelles

The progression is from information that is encoded direclty in the DNA to information that is not so encoded, and might even be completely absent.

Sidebar: Trojan Horses and the Evolution of Compilation

Dishonest translation, in which the product does not precisely mirror the original encoding, exists also in computers. The idea is known as a Trojan horse, so named after the wooden artifact from Greek mythology which contained more than was seen on the surface. In software, the term Trojan horse refers to an executable that includes code that is not a translation of the original program — instead, it is code that was added later, usually maliciously.

A good example of malicious code is computer viruses. A virus is a piece of code that is unobtrusively attached to a program, and changes its function under certain circumstances. Thus one of the main ways to detect a viral infection is to create digital fingerprints that allow one to detect when the executable has changed. An executable that no longer fits its fingerprint is most probably infected by a virus.

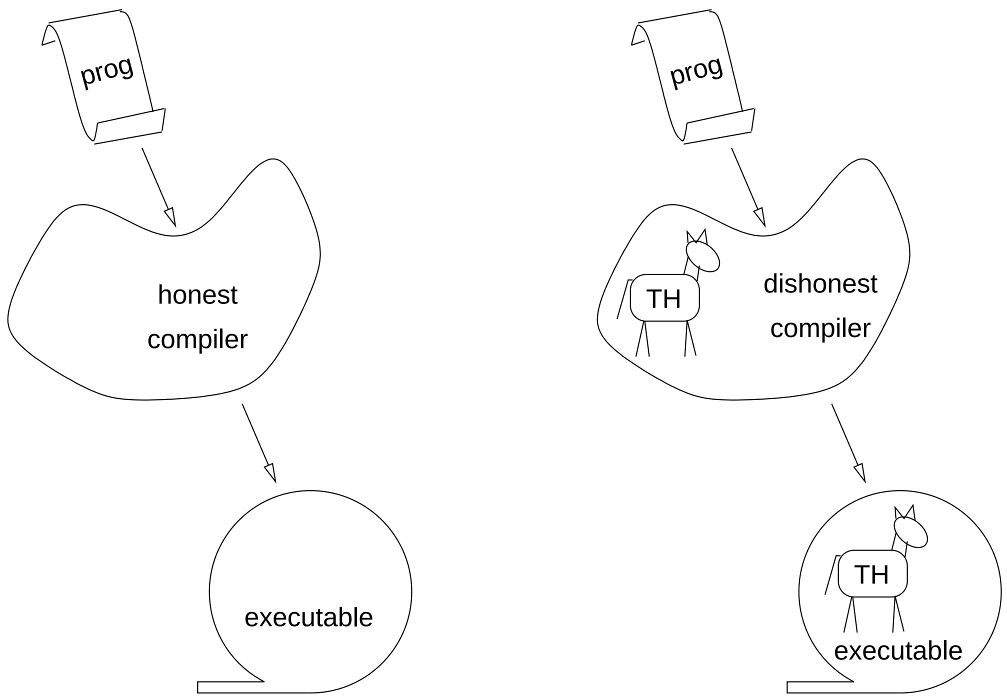


Figure 3: *A dishonest compiler can add malicious code (a Trojan horse, TH) in the process of compilation.*

But such detection can be avoided if the compiler itself inserts the virus from the outset, so the original fingerprint already includes the virus in addition to the legitimate code (Fig. 3). We can then no longer hope to be able to detect infected software. But if we suspect the compiler, we might inspect its source code, and even re-compile the compiler itself before we use it [2]. Surely we can then trust the code it produces?

The answer turns out to be no [12]. We can write a compiler that recognizes its own code, and adds the routine that infects software with the virus whenever it re-compiles itself. This routine is then erased from the source code, that is left with only legitimate translation instructions. However, when compiled by the corrupt compiler, the result will be a new corrupt compiler rather than a new honest compiler. Note that the corrupting routine no longer exists in the *code* from which the compiler is compiled — it exists only in the *compiled* compiler, and is propagated by it from one generation to the next. Thus even if we inspect the source of our application and even the source of the compiler, and recompile both of them, we still cannot be sure that the resulting executable is not infected.

Something similar to the above scenario may happen even without any malicious intent. In large software projects involving the design of new languages, it sometimes happens that after several generations of language development, it is no longer possible to re-compile the whole system from scratch.

The process is as follows. Assume the new language is called $\mathcal{L}$, with versions denoted by a superscript ($\mathcal{L}^1$, $\mathcal{L}^2$, and so on). First, a compiler is written in another language $\mathcal{S}$ for compiling language $\mathcal{L}^1$, and compiled. Call the resulting compiler $\mathcal{C}_{\mathcal{L}^1}$. Now language $\mathcal{L}^1$ itself is used to write a compiler for the next version, $\mathcal{L}^2$. $\mathcal{C}_{\mathcal{L}^1}$ is used to compile this second compiler, creating $\mathcal{C}_{\mathcal{L}^2}$. We can now use $\mathcal{L}^2$, which presumably has more features, to write an even more advanced compiler for $\mathcal{L}^3$. This goes on with compilers being written in more and mode advanced versions of the language, and being used to compile the next version.

In principle, this whole itertive process can be repeated starting from the original compiler in language $\mathcal{S}$. But this depends on all the intermediate versions of the source code being kept and catalogued in the order they were produced. If one of the intermediate versions is misplaced, we are left with a working system that we can continue to develop, but we cannot recreate it from scratch. Part of the required knowledge is contained in the operational system itself, and does not exist any more in the source code. This does in fact happen in Human-generated systems, even if perhaps it should not.

end sidebar

Chicken and Egg

A major difficulty with the notion that all the required information is contained in the DNA is that it seems that this information is useless without all the surrounding cellular machinary. While the DNA contains basic instructions on how to prepare many *components* of this machinary (namely proteins), it is unlikely to contain full instructions on how to assemble them into super-molecular structures and to create a functional cell. And some components that are not made of proteins are missing altogether. Moreover, many different cell types are possible with the same DNA, so obviously additional information is needed. Thus DNA is only meaningful with a cellular context in which it can express itself, and there is an iterative, cyclic relationship between the DNA and the context.

The obvious question when faced by such an interlinked iterative process is the chicken and egg problem: how did it all start? This question is actually more complicated than

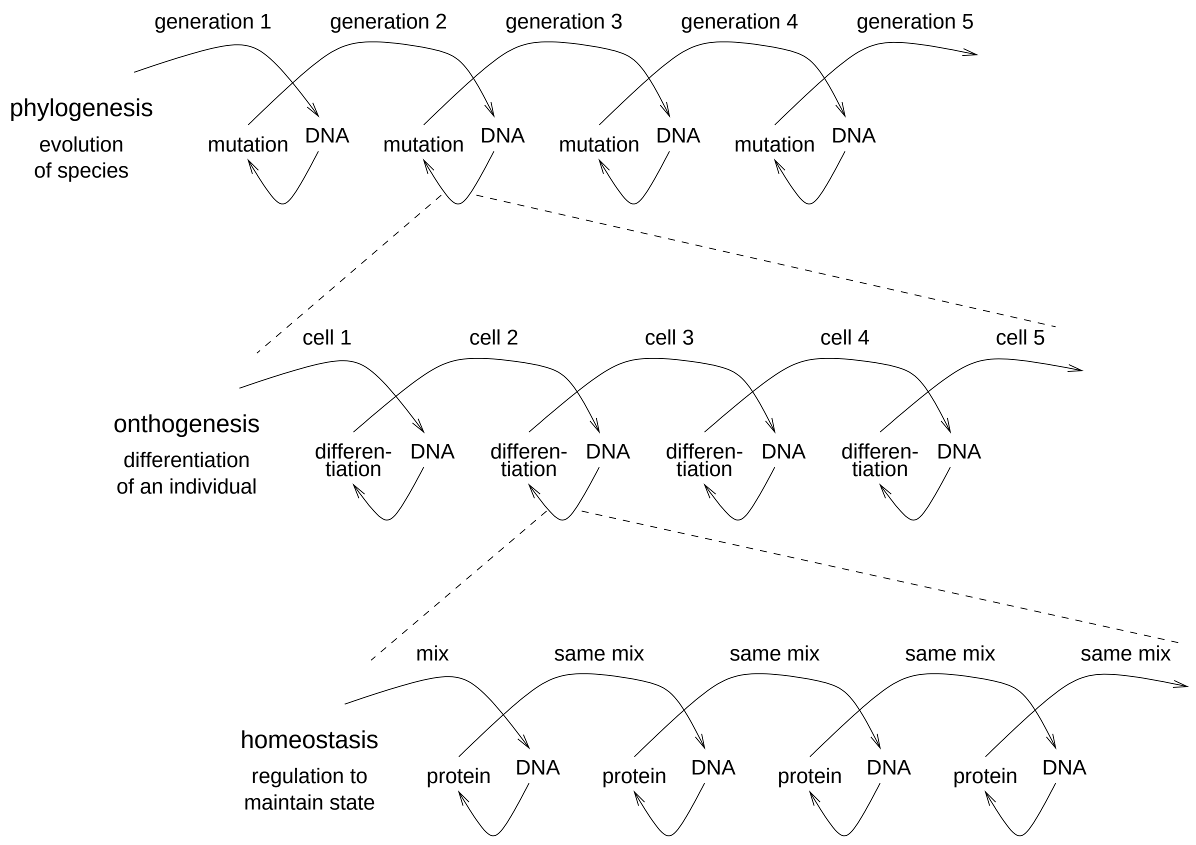


Figure 4: *Three scales in the spiral of life.*

implied above, because the feedback cycle is very tight. DNA is replicated, maintained, and protected by proteins. You can't create proteins from DNA if you do not already have proteins that assist in the transcription process (polymerases), and proteins that assist the new ones to fold correctly into their intended 3-D configuration (chaperones).

The answer to the chicken-and-egg problem is that the cycle we see today is the result of a very long spiral that represents the evolutionary process. It indeed can't be split in any given place, because each element of the cycle depends on those around it. The quest to understand the origins of life is the quest to reconstruct the spiral. This involves understanding all the processes that drive the cycle, and are not encoded in the DNA.

One of the sources of complexity in life is that actually we are talking about spirals at three different levels (Fig. 4):

1. The evolutionary scale, at which organisms reproduce and mutate leading to a divergence of species (phylogenesis),

2. The multi-cellular scale, at which cells of an organism divide and differentiate, creating the different body parts (onthogenesis), and

3. The single-cell scale, in which transcription is typically done with the goal of maintaining homeostasis (a stable state). In other words, genes are mainly transcribed to make up for proteins that have degraded.

In computer science terms, one can talk about three levels of nested parallel loops: that of multiple interacting organisms, that of multiple cells within each organism, and that of multiple proteins expressed in each cell. This is massive parallelism with very little synchrony. The complexity comes from the fact that by necessity the forces that drive all three levels of the cycle of life are embodied in the composition of the cytoplasm, and in the configuration of the cell. Fully understanding it requires a lot of details to be known.

There's still a lot to do in Biology to uderstand these and related issues. It is not all digitally encoded information. Consider a small robot entrusted with the task of reconstructing a functional cellular environment. This includes the structure of various organelles, the quantities of different proteins, their localization in the cell, and the constructs they create. How much information will the robot need in order to perform its task? Merely understanding the information encoded in the DNA may end up being the easy part.

References

[1] J. Beisson and T. M. Sonneborn, "*Cytoplasmatic inheritance of the organization of the cell cortex in Paramecium aurelia*". *Proc. Nat'l Acad. Sci.* **53**, pp. 275–282, 1965.

[2] J. M. Boyle, R. D. Resler, and V. L. Winter, "*Do you trust your compiler?*". *Computer* **32(5)**, pp. 65–73, May 1999.

[3] Y. O. Chernoff, "*Mutation processes at the protein level: is Lamarck back?*". *Mutation Research* **488(1)**, pp. 39–64, 2001.

[4] J. P. Fitch and B. Sokhansanj, "*Genomic engineering: moving beyond DNA sequence to function*". *Proc. IEEE* **88(12)**, pp. 1949–1971, Dec 2000.

[5] E. Jablonka, M. Lachmann, and M. J. Lamb, "*Evidence, machanisms and models for the inheritance of acquired characters*". *J. Theor. Biol.* **158**, pp. 245–268, 1992.

[6] J. Maynard Smith, "*Models of a dual inheritance system*". *J. Theor. Biol.* **143**, pp. 41–53, 1990.

[7] G. L. G. Miklos and G. M. Rubin, "*The role of the genome project in determining gene function: insights from model organisms*". *Cell* **86(4)**, pp. 521–529, Aug 1996.

[8] D. L. Nanney, "*Cortical patterns in cellular morphogenesis*". *Science* **160**, pp. 469–502, May 1968.

[9] A. Razin and H. Cedar, "*DNA methylation and genomic imprinting*". *Cell* **77**, pp. 473–476, 1994.

[10] M. G. Roth, "*Inheriting the Golgi*". *Cell* **99(6)**, pp. 559–562, Dec 1999.

[11] T. R. Serio and S. L. Lindquist, "*Protein-only inheritance in yeast: something to get [PSI$^+$]-ched about*". *Trends Cell Biol.* **10**, pp. 98–105, Mar 2000.

[12] K. Thompson, "*Reflections on trusting trust*". *Comm. ACM* **27(8)**, pp. 761–763, Aug 1984.

[13] P. D. Ward and D. Brownlee, *Rare Earth.* Copernicus/Springer-Verlag, 2000.

[14] I. Wilmut, A. E. Schneike, J. McWhir, A. J. Kind, and K. H. Campbell, "*Viable offspring derived from fetal and adult mammalian cells*". *Nature* **385(6619)**, pp. 810–813, Feb 1997.

[15] M. P. Yaffe, "*The machinery of mitochondrial inheritance and behavior*". *Science* **283**, pp. 1493–1497, Mar 1999.