Language, Literature and Culture
2018; 1(1): 23-29
http://www.aascit.org/journal/llc

AASCIT American Association for Science and Technology

Linguistic Thinking on DNA as Programming Language

Yanqiu Tong[1], Tianyi Zhang[2], Yufei Deng[2], Yang Song[3]

[1]Department of Humanity School, Chongqing Jiaotong University, Chongqing, China
[2]Department of Foreign Language School, Sichuan International Studies University, Chongqing, China
[3]Department of Device Chongqing Medical University, Chongqing, China

Email address

yqtong@126.com (Yanqiu Tong)

Citation

Yanqiu Tong, Tianyi Zhang, Yufei Deng, Yang Song. Linguistic Thinking on DNA as Programming Language. *Language, Literature and Culture.* Vol.1, No. 1, 2018, pp. 23-29.

Received: March 20, 2018; **Accepted**: April 2, 2018; **Published**: May 16, 2018

Abstract: Recent years lots of research have pay focus on multitude of formal languages and systems applied to biology, thus gaining insight into the biological systems under study through analysis and simulation. And in information processing field, well-designed DNA circuits have been implemented in DNA by using strand displacement as their main computational mechanism. The purpose of this paper is to discuss the linguistic character of programming language for designing and simulating DNA strand displacement. In which strand displacement is the main computational mechanism. A compared method was introduced for compare with human language and DNA. And the genetic mechanism was described as a magical language that can be considered as human language. In conclusion, DNA strand displacement as programming language has the property of human language.

Keywords: DNA, Language, Strand Displacement

1. Introduction

Nucleic acids have a number of desirable properties like language. DNA sequences can be precisely organized in order to describe distinct meanings. DNA helix was founded by Watson–Crick and the pairing rule can be used to interact between specific molecules at well-defined rates [1, 2]. Previous efforts in designing biochemical circuits with DNA have tended to make use of additional restriction enzymes, or structural features such as hairpins within the molecules to perform computation [3-5]. While this allows the implementation of somewhat ingenious molecular devices [6, 7], simpler designs have recently been proposed for the construction of large-scale, modular circuits. In particular, a range of information-processing circuits have recently been implemented in DNA by using strand displacement as the main chemical process to perform computation. Examples include various digital logic circuits together with catalytic signal amplification circuits that function as efficient molecular detectors [8, 9]. The use of DNA strand displacement to perform computation enables the construction of simple, fast, modular composable and robust circuits, as demonstrated in Zhang et al. (2007).

A range of modelling approaches have also been developed for DNA computation [10]. Computing techniques got wider acceptance due to biological complexity and computational properties of DNA [11]. Such operations can effectively model Adleman's experiment [12], in which DNA was used to compute a Hamiltonian path in a graph. Other examples include Watson–Crick automata, which are the automata counterpart to sticker systems, insertion–deletion systems, which contain operations for inserting and deleting DNA sequences, and splicing systems, which can be physically implemented with the help of restriction enzymes. A more recent review of modelling approaches is presented in Amos (2005), together with their corresponding physical implementations.

2. DNA Is a Language for Biochemical Systems

2.1. Basic Grammer

Simple examples. We present a language for DNA strand displacement by means of simple examples, together with their corresponding graphical representation. The design of

the language is motivated by the assumptions outlined in Zhang et al [13]. Examples of DNA molecules are presented below, where parallel composition (|) denotes the presence of multiple molecules next to each other. Figure 1 show the ruler of DNA molecule computing.

Figure 1. *The ruler of DNA molecule computing.*

The molecule 1: 2 represents a lower strand of DNA, where the 30 end of the strand is assumed to be on the left, as indicated by an arrowhead in the graphical representation. The strand is divided into domains, which correspond to short DNA sequences. The domains are represented by numbers 1 and 2, where each number represents a distinct domain. The DNA sequences of distinct domains are assumed to be sufficiently different that they do not interfere with each other. The red domain 1 represents a toehold domain, while the black domain 2 represents an ordinary specificity domain. The colour is merely an annotation, since the length of the domain sequence is sufficient to determine its type. Toehold domains are very short sequences, generally between 4 and 10 nucleotides in length, that enable one DNA strand to bind to another. Since the sequence is short, the two strands will quickly unbind from each other in the absence of further interaction along neighbouring domains. The molecule, 1 2. represents an upper strand of DNA, where the 30 end of the strand is assumed to be on the right. The strand consists of two domains that are complementary to domains 1 and 2, where two domains are complementary if their respective sequences are Watson–Crick complementary. We denote 1: 2 as a lower strand and, 1 2. as an upper strand in order to emphasize the complementarity between strands. Two complementary strands 1: 2 and, 1 2. can hybridize along their complementary domains to form a double-stranded molecule [1 2]. A molecule can also consist of multiple upper strands bound to a single lower strand. For example, [1 2]: [3 4] consists of upper strands, 1 2. and, 3 4. bound to a single lower strand 1:2:3:4. There can also be gaps between bound upper strands, as in the molecule [1 2]: 3: [4 5], where domain 3 of the lower strand is unoccupied.

2.2. Tools

Compared to the other languages for biochemical modelling mentioned in the introduction, CBS is unique in its combination of two features: it explicitly models reactions rather than individual agents as in process calculi, and it does so in a compositional manner. BIOCHAM, for example, also models reactions explicitly using a very similar syntax to that of CBS, but it does not have a modular structure. Whether the explicit modelling of reactions is desirable or not depends on the particular application. Systems which are characterized by high combinatorial complexity arising from complex formations are difficult or even impossible to model in CBS and LBS; an example is a model of scaffold formation which considers all possible orders of subunit assembly, where one may prefer to use Kappa or BioNetGen. But for systems in which this is not an issue, or where the combinatorial complexity arises from modifications of simple species (which CBS and LBS deal with in terms of match variables), the simplicity of a reaction-based approach is attractive. It also corresponds well to graphical representations of biological systems, as we have seen with the yeast pheromone pathway example.

To our knowledge, no other languages have abstractions corresponding to the pattern expressions and nested declarations of species and compartments of LBS. The notion of parameterized modules is however featured in the Human-Readable Model Definition Language [14], a draft textual language intended as a front-end to the Systems Biology Markup Language (SBML) [15]. The modules in this language follow an object-oriented approach rather than our functional approach, but there is no notion of subtyping or formal semantics. Tools for visualizing LBS programs and, conversely, for generating LBS programs from visual diagrams, are planned and will follow the notation of [16, 17] or the emerging Systems Biology Graphical Notation (SBGN). We also plan to use LBS for modelling large scale systems such as the EGFR signal pathway [18], although problems with interpretation of the graphical diagrams are anticipated. While some parts of the EGFR map are well characterized by modules, others appear rather monolithic and this is likely to be a general problem with modular approaches to modelling in systems biology. In the setting of synthetic biology, however, systems are programmed rather than modelled, so it should be possible to fully exploit modularity there. With respect to language development, it is important to achieve a better understanding of homomers, enabling a commutative pattern composition operation.

2.3. Difference from Computer Language

In the perspective of genetic mechanism, DNA is a kind of language, but it different from computer language. DNA is more advanced than human language. The reason is simple, DNA is extremely dynamic (unstable) and computer language is relatively static (stable).

We don't want to be ready to accept either course language, factors of instability, it will be based virus hackers. If an instruction can be output for several kinds of results, then we will need a computer output? The estimation results is the crash. DNA dynamic refers to DNA genetic mechanism itself is constantly modify their own and evolve. So it is a low-level software; DNA is a senior software. From

low to high, the genetic evolution mechanism itself constantly, mutation, is selected, to modify their own. Species from low to high evolutionary tree, if you look at their DNA, you can see the software update history. This history is not linear, but as a family tree, sometimes develop independent branches. For example, in insects, larval Drosophila salivary gland cells., chromosome structure will produce a multiple called polytene chromosome, is one thousand times of chromosome replication then tied together, scholars believe that this structure is helpful to a large number of secretory protein. However this structure only in flies there. Other insects have their own similar but different DNA structure in a large number of secretory protein.

The genetic mechanism is a magical language that can be considered as language as follow:

(1) Missense mutation

Missense mutation or substitution refers to a change in one amino acid in a protein, arising from a point mutation in a single nucleotide. Missense mutation is a type of nonsynonymous substitution in a DNA sequence, see figure 2.

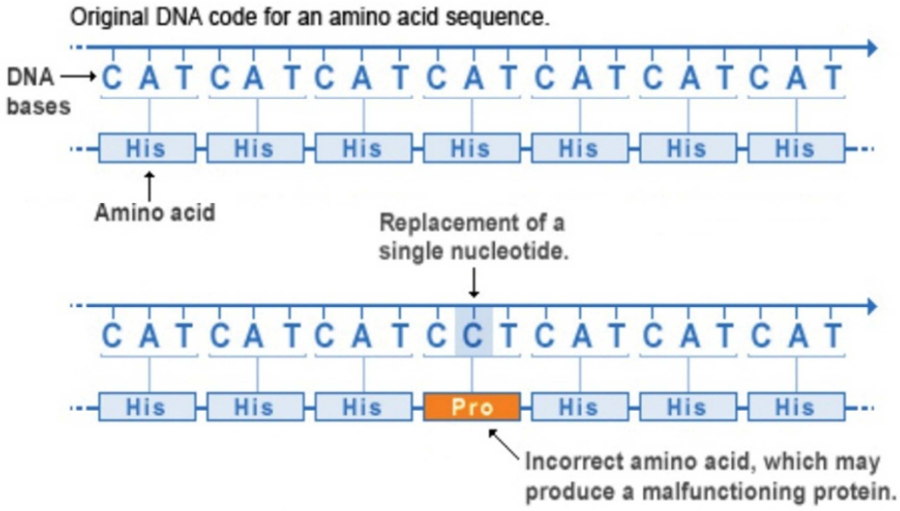

Figure 2. *Missense mutation.*

In this example, the nucleotide adenine is replaced by cytosine in the genetic code, introducing an incorrect amino acid into the protein sequence.

(2) Nonsense mutation

In genetics, a point-nonsense mutation is a point mutation in a sequence of DNA that results in a premature stop codon, or a point-nonsense codon in the transcribed mRNA, and in a truncated, incomplete, and usually nonfunctional protein product, see figure 3.

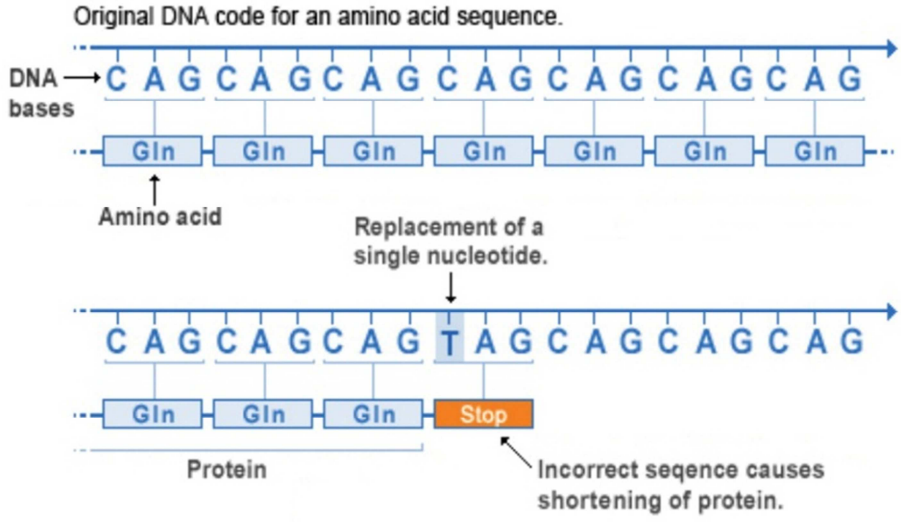

Figure 3. *Nonsense mutation.*

In this example, the nucleotide cytosine is replaced by thymine in the DNA code, signaling the cell to shorten the protein.

(3) Insertion mutation

In genetics, an insertion (also called an insertion mutation) is the addition of one or more nucleotide base pairs into a

DNA sequence. This can often happen in microsatellite regions due to the DNA polymerase slipping. Insertions can be anywhere in size from one base pair incorrectly inserted into a DNA sequence to a section of one chromosome inserted into another, see figure 4.

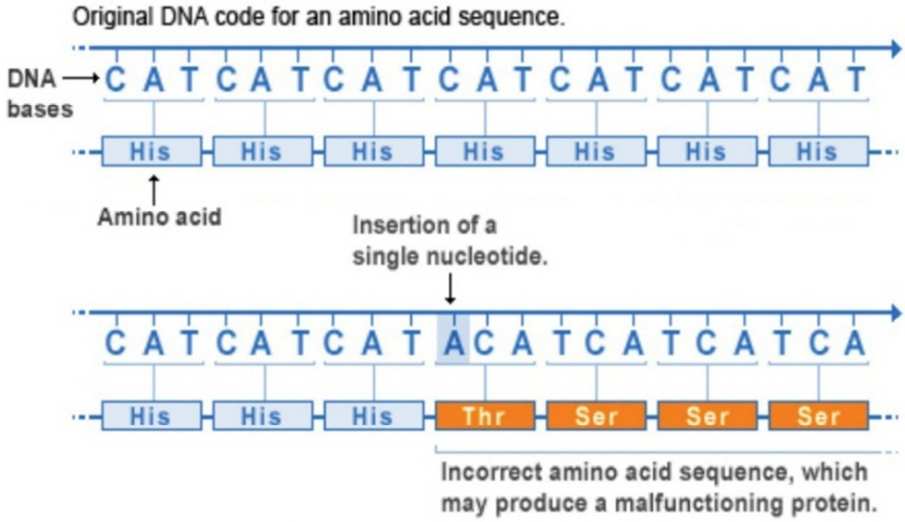

Figure 4. *Insertion mutation.*

In this example, one nucleotide is added in the DNA code, changing the amino acid sequence that follows.

(4) Deletion mutation

In genetics, a deletion (also called gene deletion, deficiency, or deletion mutation) (sign: Δ) is a mutation (a genetic aberration) in which a part of a chromosome or a sequence of DNA is lost during DNA replication. Any number of nucleotides can be deleted, from a single base to an entire piece of chromosome (Lewis, R. (2004). Human Genetics: Concepts and Applications (6th ed.). McGraw Hill. ISBN 0072951745.), see figure 5.

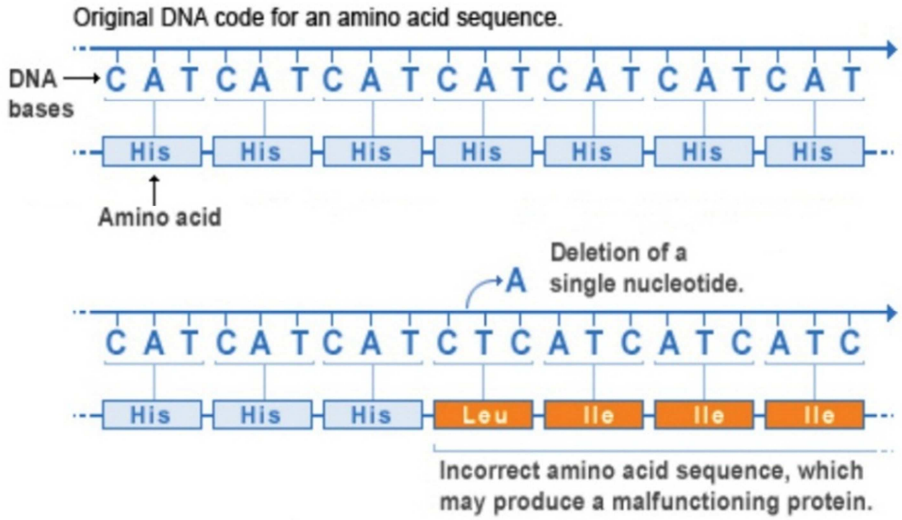

Figure 5. *Deletion mutation.*

In this example, one nucleotide is deleted from the DNA code, changing the amino acid sequence that follows.

(5) Duplication mutation

In biology, a mutation is a change in the genetic material. This means changes to the DNA or to the chromosomes which carry the DNA. These changes are heritable (can be passed on to the next generation) unless they have lethal effects, see figure 6.

(6) Frameshift mutation

A frameshift mutation (also called a framing error or a reading frame shift) is a genetic mutation caused by indels (insertions or deletions) of a number of nucleotides in a DNA sequence that is not divisible by three. Due to the triplet nature of gene expression by codons, the insertion or deletion can change the reading frame (the grouping of the codons), resulting in a completely different translation from the original, see figure 7.

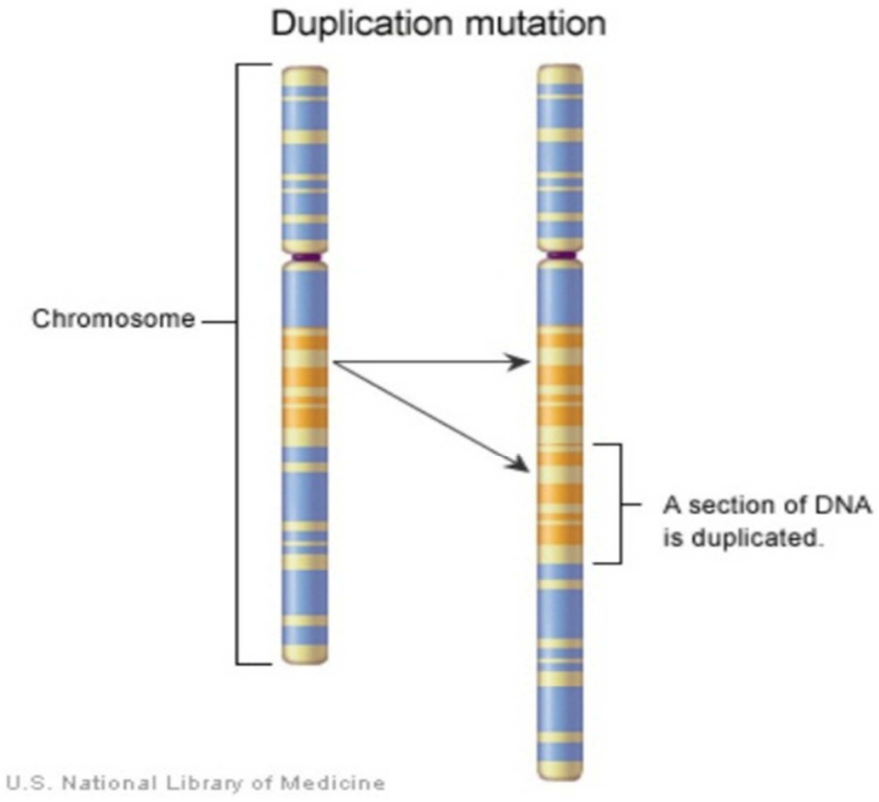

Figure 6. *Duplication mutation.*

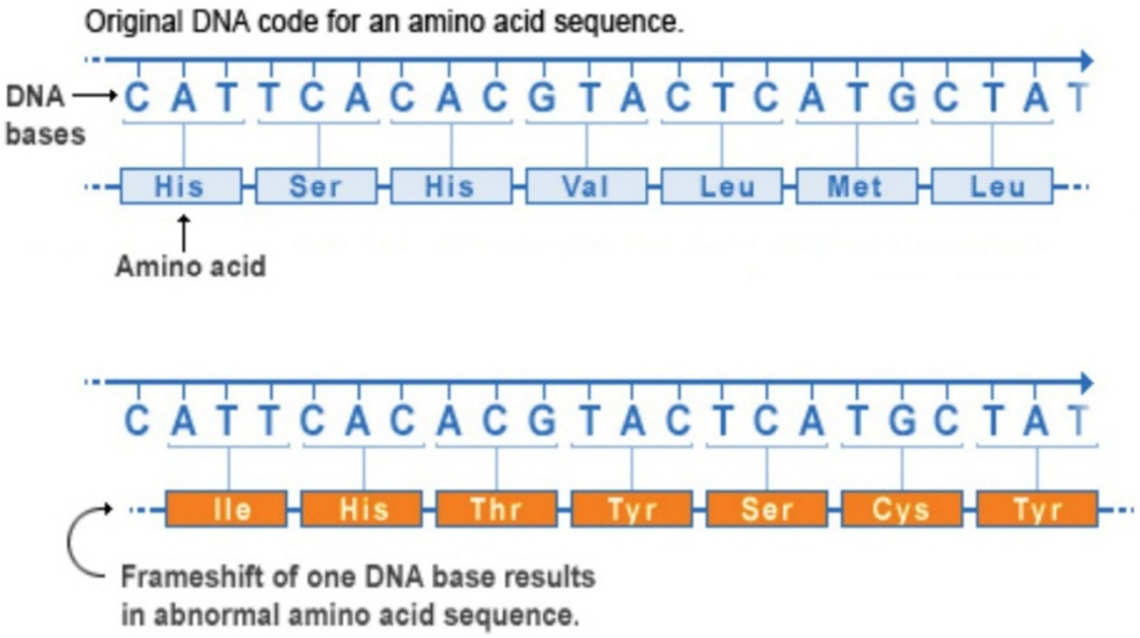

Figure 7. *Frameshift mutation.*

(7) Repeat expansion mutation

Trinucleotide repeat expansion, also known as triplet repeat expansion, is the DNA mutation responsible for causing any type of disorder categorized as a trinucleotide repeat disorder. These are labelled in dynamical genetics as dynamic mutations [19]. Triplet expansion is caused by slippage during DNA replication, also known as "copy choice" DNA replication [20], see figure 8.

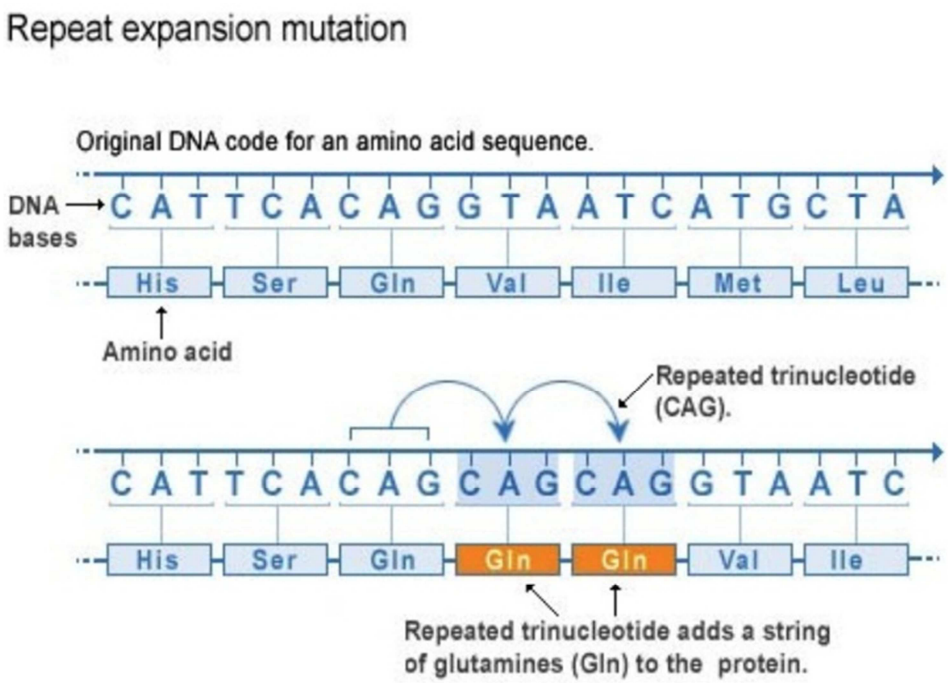

Figure 8. *Repeat expansion mutation.*

DNA is transcribed into RNA to express protein, this process, several stages, each stage has a variety of N repair Decoration and regulation, is an extremely huge control system network changes, you can imagine each regulation and modification can be used as the basis of evolution. The 4 DNA molecule itself is jumping. This is a big difference with the computer language. DNA is material, a long chain of concentration is large enough it can touch. The nucleic acid chain is relatively stable: it has a lot of uncertainties of the sequence. Some sequence will copy themselves, then the inside of the gene has a copy, then you have the mutation, you have a backup. Some sequences, like transposon transposable element, will jump off the original position, inserts itself to another location. 5 genes have own defense mechanism. Just like computer antivirus software, but he is built in, is your original code inside anti-virus code. Between 6 different grades of DNA language is interaction Dynamic. So now you should be feeling, DNA why n times than our language. He is like a set of reasonable legal and political system, it is continuously revised - to improve their own. And our computer language, most of the time is to us, the programming, the problems found, and then go to the changes. If there is such a software, his code is automatically found, when you use the automatic updates, so strengthen such language, is close to the DNA language.

3. Conclusion

The formation and design of DNA is completely random, the evolution of DNA is by different levels of mutation and selection based on the environment, means to sacrifice a lot of individuals to obtain a more advanced individual. I am not familiar with the history of evolution, I know the life is spent tens of millions to determine the expression and basis of "grammar". DNA function is the regulation of inefficient. DNA is extremely complex, on another level, friends can have a more simple procedure to express the same information. DNA protein is a lot of wrong. Of course, there is a mistake, there is progress, only evolution the space.

The programming language is the product of human wisdom. Human learn the principles of nature, study mathematics, subjectively update and improve program. It is guarantee that the programs developed are the most efficient, that is, the most efficient. Languages and programs designed by humans try to avoid errors. If we want, it is perfectly possible to add random errors to the program. But it is extremely difficult to make the program "copy" itself.

Funding

This work was supported by Chongqing Kitty Hawk Plan Research Project (CY170701), The Humanities and Social Sciences project of the Chongqing Municipal Education Committee (16SKGH022), Chongqing Federation of Social Sciences (2017ZDYY10), Chongqing Science and Technology Commission (cstc2016jcyjA2034).

References

[1] Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z. & Shapiro, E. 2001 Programmable and autonomous computing machine made of biomolecules. Nature 414, 430–434.

[2] Benenson, Y., Adar, R., Paz-Elizur, T., Livneh, Z. & Shapiro, E. 2003 DNA molecule provides a computing machine with both data and fuel. Proc. Natl Acad. Sci. USA 100, 2191–2196.

[3] Sakamoto, K., Gouzu, H., Komiya, K., Kiga, D., Yokoyama, S., Yokomori, T. & Hagiya, M. 2000 Molecular computation by DNA hairpin formation. Science 288, 1223–1226.

[4] Benenson, Y., Gil, B., Ben-Dor, U., Adar, R. & Shapiro, E. 2004 An autonomous molecular computer for logical control of gene expression. Nature 429, 423–429.

[5] Yin, P., Choi, H. M. T., Calvert, C. R. & Pierce, N. A. 2008 Programming biomolecular self-assembly pathways. Nature 451, 318–322.

[6] Yurke, B. & Mills Jr, A. P., 2003 Using DNA to power nanostructures. Genet. Program. Evolvable Mach. 4, 111–122.

[7] Venkataraman, S., Dirks, R. M., Rothemund, P. W. K., Winfree, E. & Pierce, N. A. 2007 An autonomous polymerization motor powered by DNA hybridization. Nature Nanotechnol. 2, 490–494.

[8] Seelig, G., Soloveichik, D., Zhang, D. Y. & Winfree, E. 2006 Enzyme-free nucleic acid logic circuits. Science 314, 1585–1588.

[9] Zhang, D. Y., Turberfield, A. J., Yurke, B. &Winfree, E. 2007 Engineering entropy-driven reactions and networks catalyzed by DNA. Science 318, 1121–1125.

[10] Jin Xu, Xiaoli Qiang, Kai Zhang, Cheng Zhang, Jing Yang, A DNA Computing Model for the Graph Vertex Coloring Problem Based on a Probe Graph, Engineering, 2018, 2, In Press.

[11] Sreeja Cherillath Sukumaran, Misbahuddin Mohammed, PCR and Bio-signature for data confidentiality and integrity in mobile cloud computing, Journal of King Saud University - Computer and Information Sciences, 2018, 3, In Press.

[12] Adleman, L. M. 1994 Molecular computation of solutions to combinatorial problem. Science 226, 1021–1024.

[13] Zhang, D. Y., Turberfield, A. J., Yurke, B. &Winfree, E. 2007 Engineering entropy-driven reactions and networks catalyzed by DNA. Science 318, 1121–1125.

[14] Bergmann, F., Sauro, H.: Human-readable model definition language (first draft, revision 2). Technical report, Keck Graduate Institute (2006).

[15] Hucka, M., et al.: The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics 19 (4), 524–531 (2003).

[16] Kitano, H., et al.: Using process diagrams for the graphical representation of biological networks. Nat. Biotechnol. 23 (8), 961–966 (2005).

[17] Moodie, S. L., et al.: A graphical notation to describe the logical interactions of biological pathways. Journal of Integrative Bioinformatics 3 (2) (2006).

[18] Oda, K., et al.: A comprehensive pathway map of epidermal growth factor receptor signaling. Molecular Systems Biology (2005).

[19] Richards RI, Sutherland GR (1997). "Dynamic mutation: possible mechanisms and significance in human disease". Trends Biochem. Sci. 22 (11): 432–6.

[20] Salinas-Rios V, Belotserkovskii BP, Hanawalt PC (2011). "DNA slip-outs cause RNA polymerase II arrest in vitro: potential implications for genetic instability". Nucleic Acids Res. 39 (15): 1–11.