DNA – «PROGRAMMING LANGUAGE OF LIFE»

R. Hofestädt

Bielefeld University, e-mail: hofestae@techfak.uni-bielefeld.de

During the last decades methods of molecular biology are able to identify and sequence different gene functional units (DNA-units). Most of these functional units are analysed in syntax (sequence) and semantic (metabolic function). This kind of information is represented by different molecular databases and information systems, which are distributed across the world (internet). Based on this knowledge it is now possible to discuss the old and still open question if DNA can be interpreted as a programming language. In this paper we will show that the DNA can be interpreted as a programming language in the sense of computer science.

Key words: DNA functional units, formal language, programming language, data, control instruction, basic instruction, operon, gene regulation, language of life and synthetic biology.

Introduction

Regarding cellular processes two different and fundamental levels of genetic languages can be identified. The polypeptide level is coded by 20 different amino acids, which represent the alphabet of this formal language, and the polynucleotide level represented by different nucleic acids (A,T,G,C {U}). The fundamental level is the polynucleotide level with it's DNA, which represents the functional units of the metabolism. Therefore, the alphabet of the fundamental language is given by X = {A,T,G,C (U)} and the DNA is a word over this alphabet. Furthermore, specific "programming language of life" is $L \subseteq X^*$ (where * is the star operator and defines all words over X). During the 60ties Ratner (Ratner, 1977) presented the idea of a genetic language that represents different levels: codon, cistron, scripton, replicon, segregon and genome. This paper will focus on the level of functional units of the DNA, which is defined by Ratners cistron level. The idea of our interpretation will be based on two assumptions. The first is that for bacteria and virus most of the DNA/RNA-units are known today. The second is that most of the known DNA-units are universal, which means that besides the universality of the genetic code the universality of DNA-units, which includes the universality of the DNA language, can be suggested. Additionally, this paper will show that analyzed DNA-units can be interpreted as a programming language. We will present this interpretation in two steps: 1) the specification of the DNA-units in syntax (nucleotide sequence) and semantic (function) and 2) the proof that the DNA-units represent fundamental mechanisms of a computer programming language.

DNA-units

At the beginning we have to discuss the question if most of the relevant DNA-units are already analyzed. Regarding the sequenced and analyzed genomes including latest molecular knowledge we can assume:

1. Most of the DNA-units are known (Knippers, 2006).

2. For most of these units the function is known and seems to be universal.

The syntax of the DNA-units can be specified using specific Chomsky-type-2 grammars as shown in (Hofestädt, 2007). Similar to the computer programming languages the semantic is described informal using the natural language. A list of the most relevant DNA-units is shown in Table 1.

Basic features – programming language

Regarding the v. Neumann computer system architecture (Burks *et al*., 1946), which is still the

Table 1

Selection of relevant DNA-units (Knippers, 2006)

DNA-unit	Comment
Intron	sub-sequence of the structure gene
Exon	sub-sequence of the structure gene
Leader	sub-sequence of specific structure gene
Structure gene	input sequence for the protein synthesis process
Spacer	separator sequence of genes
Palindrome	antidromic DNA sequence
Terminator	end of the transcriptional unit
Pribnow-Box	subsequence of the promoter to specify promoter affinity
Promoter	start point of the transcriptional unit
Operator	sequence of the gene regulation process
Regulator	specific structure gene
Operon	unit of the protein synthesis process
Telomer	specific sequence of the end of the chromosome
Origin	startpoint of the DNA-Polymerase
Segregon	heredity unit
IS-Element	dynamical structure of the genome
Transposon	dynamical structure of the genome
Virus-DNA-RNA	dynamical structure of the genome
Enhancer	controls the promoter affinity
Overlapping gene	specific structure gene
Homeotic gene	operon, which shows the modularity of the genome

kernel of our computer, we are able to detect the following fundamental features of a programming language:

F1. ***Data type*** (at least one is sufficient): Computer instructions can modify data so that at least one data type (e. g. integer) must be available.

F2. ***Instruction:*** Each computer/computer language is offering a set of instructions, which can modify data (e. g. add, multiply, and etc.).

F3. ***Control instruction:*** Specific instruction which controls the order of the next executable instruction of the program.

F4. ***Punctuation mark:*** Begin and end symbol (word) is defined.

Besides the numeric/logical instructions, which will modify the program data, the control instructions are fundamental. Regarding all control instructions we can differrentiate between three fundamental classes:

C1: Composition S1; S2;...; Sn

The semicolon denotes the following operator. The semantic of this operator is that instruction (S_{i+1}) will be executed after execution of instruction (S_i).

C2: If-Instruction (If B then S)

Let S be an instruction and B a condition, which can be true/false. Instruction S will be executed if and only if condition B is true – otherwise S will not be executed.

C3: While-instruction (While B do S)

Let S be an instruction and B a condition, which can represent the value true/false. The meaning is that the instruction S will be executed as long as B is true.

The theoretical model of the v. Neumann architecture is the so-called Turing machine (Hopcroft, Ulman, 1979), which belongs to the class of the universal computational concepts. That means each problem, which is intuitive computable, can be solved using the adequate Turing machine

(v. Neumann computer). Regarding the DNA-units, which are shown in Table 1, we can define the basic instruction of the DNA language and the activation of this instruction. A DNA-unit is called basic instruction, if at least one promoter and one terminator are included. If a ***basic instruction*** will pass the biosynthetic process, this is called ***activation*** of the basic instruction.

An operon can be interpreted as a basic instruction, which includes structure genes, operator genes, one or more promoter genes and a terminator gene. The activation of a basic instruction includes fundamental metabolic processes like transcription and translation (Knippers, 2006). It is similar to the instruction execution process of a computer system.

Interpretation

Assume that the DNA is the genetic program of the cell. In that case the cytoplasm can be interpreted as the data type which represents metabolites. Metabolites can be modified by different biochemical reactions. Therefore, we can assume that the data type (metabolite) is available (see F1). Enzymes can catalyze biochemical substances so that a substrate will be modified into a product catalysed by a specific enzyme. Therefore, instructions are chemical reactions caused by enzymes, which are presented by structure genes (see F2). Furthermore, structure genes are controlling the metabolism indirectly. Regarding specific cells we can see specific genes, which are active during specific time periods. This behaviour shows that specific DNA-units control the activity of genes, which can be interpreted as the control instruction (see F3). Finally, the DNA-unit, which is called telomer, can be interpreted as the punctuation mark of this system (see F4). Defining and regarding specific operons it is possible to show that the control instructions C1 – C3 can be simulated by gene controlled regulatory networks. Composition of basic instructions can be interpreted as a sequence of basic instructions represented by structure genes or operons separated by spacer units (Fig. 1).

Furthermore, we can show that an operon can be interpreted as an If-instruction (see C2). Therefore, we focus on the operon L14 of *E. coli*, which regulates its own synthesis. The mechanism of this regulation process can be illustrated (Fig. 2).

Under this interpretation the boolean value of condition B will be specified by the state of the operator, which can be true (*Operator_X* gene is free) or false (*Operator_X* is blocked by the repressor). Under this interpretation the instruction **If B then S** is simulated, because the operon will

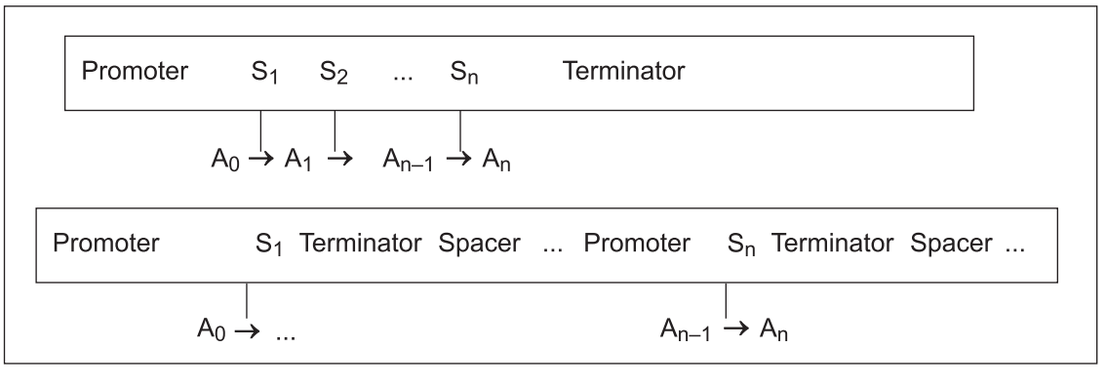

Fig. 1. Composition of basic instructions of the DNA language.

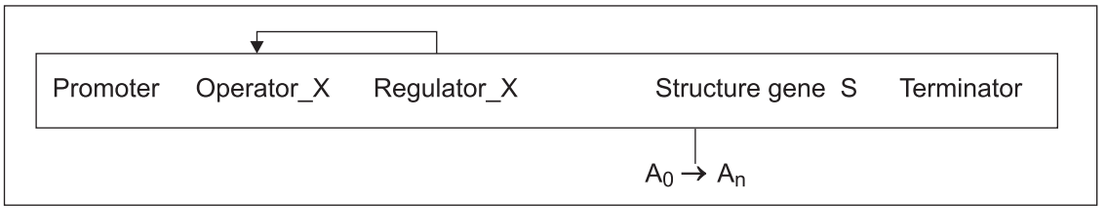

Fig. 2. Abstract representation of the operon L14 (Knippers, 2006), which can be interpreted as the If-instruction.

be blocked by itself after activation of this basic instruction (operon). The synthesis of *Structure gene S* will realise the instruction S and the synthesis of *Regulator_X* will block the synthesis of operon L14. Regarding the example of the illustrated operon L14 the theoretical extension to the **While instruction** can be realised. Deleting the *Regulator_X* gene, which is inside of the operon L14, will produce this effect. Therefore, we will discuss the structure and function of the *Tryptophan* operon, which shows this effect (Fig. 3).

Regarding the *Tryptophan* operon, we can see a composition of structure genes (S) and the boolean value of condition B can be true (Operator is free) or false (Operator is blocked).

As long as the operon represents the state true the basic instruction is activated. It will be activated until the operator gene will be blocked. This interpretation shows the «While operator» (see C3). Furthermore, the telomere sequence (chromosome) and the terminator/promoter sequence (cistron level) can be interpreted as the begin- and end-mark (F4).

Features of DNA-languages

Regarding the semantic of the analyzed DNA-units we can identify the following features of this language. First of all the genome is modular organized like most of the programming languages. The modular organization can be shown regarding the semantic of the homeotic genes (Watson *et al*., 1985). These genes are controlling a battery of genes and can be interpreted as a function, procedure or module in the sense of programming languages. Regarding the semantic of the promoter we can see that the basic element will be activated based on a probability value, which is defined by the specific sequence of the promoter sequence. These sequences specify the so-called promoter affinity of the operon (Knippers, 2006). One more interesting and complex feature is the dynamic behavior of DNA-units, which is shown for example by transposons and IS-elements (Watson *et al*., 1985). The semantic of transposons and IS-elements is that DNA sequences can change their localization inside the chromosome (program). Furthermore, changing the localization other DNA-units can be destroyed or modified. Until now the semantic of these dynamic DNA-units is not known exactly. Parallelization is one more feature of molecular processing. The opposite to von Neumann computer basic molecular instructions can be activated in parallel. This concept is called dataflow concept and the semantic is that each basic instruction will be executed if all biochemical conditions are satisfied (operator genes are free and RNA-polymerase is available etc.). Regarding these features we can say the programming language of life is much more complex than a v. Neumann computer language. Regarding theoretical models of computation discussed in theoretical computer science we can see parallel machines (Fortune, Wyllie, 1978), probabilistic Turing machines (Gill, 1977), and hardware modification machines (Cook, 1980). However, there is no discussion about a dynamic-parallel-probabilistic-modular Turing machine model.

Summary

The key idea of this paper is to show that the DNA can be interpreted as a programming language based on the level of analyzed functional DNA-units. Therefore, we presented a subset of well-known DNA-units and extracted the main features of a von Neumann programming language. Our paper shows that DNA-units can be interpreted as a programming language based on the level of DNA-units, which was called cistron level by the fundamental definition of Ratner (Ratner, 1977). Therefore, metabolites can be interpreted

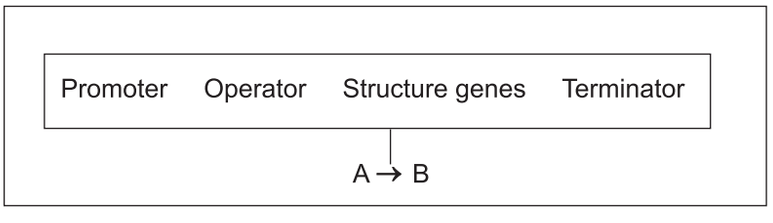

Fig. 3. The *Tryptophan* operon (Knippers, 2006) represents: Promoter, terminator, operator and a composition of structure genes: *trpE*, *trpD*, *trpC*, *trpB*, *trpA*.

as the fundamental data type and operons which synthesize enzymes that can be interpreted as data modifying instructions. Specific operons can also be interpreted to simulate the fundamental computer instructions as: composition, If- and While-statements. Overall section 4 showed that the fundamental structures of a programming language are represented by basic DNA-units. Therefore, the DNA-units represent the fundamental language of life. Further discussions showed that this language is representing complex language structures as features of parallel, probabilistic, dynamic and modular computing. Regarding computational models of theoretical computer science we can see no model, which represents such a complex computational mechanism today.

Thinking about the post-genome era it is the main idea of this work to study and understand this kind of complex languages (computational methods), which we could identify in this paper. Based on this work and the understanding of such methods and concepts it will be possible to specify the language of life in detail, which will represent the framework for the realization of the ideas of synthetic biology.

References

Burks A., Goldstine H., Neumann von J. Preliminary Discussion of the Logical Design of an Electronical Computing Instrument // Research Report. Institute of Advanced Study, Princeton, 1946.

Cook S. Towards a complexity theory of synchronous parallel computation // L'Enseignement Mathematique. 1980. 27. P. 99–124.

Fortune S., Wyllie J. Parallelism in Random Acess Machines // Proc. 10th ACM Symp. on Theory and Computing, 1978. P. 114–118.

Gill J. Computational complexity of probabilistic Turing Machines // SIAM J. of Computing. 1977. 6. P. 675–695.

Hofestädt R. Extended Backus-System for the representation and specification of the genome // J. of Bioinformatics and Computational Biol. 2007. 5-2(b). P. 457–466.

Hopcroft J.E., Ullman J.D. Automate Theory, Languages, and Computation, Addison-Wesley Publ. Co., London – Amsterdam Don Mills, Ontario – Sydney 1979.

Knippers R. Molekulare Genetik, Stuttgart; N.Y.: Georg Thieme Verlag, 2006.

Ratner V. Molekulargenetische Steuerungssysteme. Stuttgart: Gustav Fischer Verlag, 1977.

Watson J.D., Tooze J., Kurtz D. Rekombinierte DNA. Spektrum der Wissenschaft, Heidelberg 1985.