On bio-design of Argo-machine

Andrew Kuznetsov, Mark Schmitz, Kristian Müller

Institute of Biology III, Albert-Ludwig's University Freiburg,
Schaenzlestrasse 1, D-79104 Freiburg, Germany
andrei.kouznetsov@biologie.uni-freiburg.de
http://omnibus.uni-freiburg.de/~kouznet/

Abstract

We describe a non-deterministic abstract machine that searches according to oracle words in the design space to fit with its environment, cuts, transposes and pastes a set of tapes. On the basis of this construct a distributed concurrent computation with DNA by bio-molecules is proposed. For this purpose we are developing a site-guidable nuclease. Labeled oligonucleotides or PNA are used as an input. Two kinds of computation are discussed: 1) *in vitro* in a thermo-cycler or 2) *in vivo* after the transgene installation into living cells.

Key words: *biocomputing, guided nucleases, evolved systems, orthogonal life*

1. Introduction

A DNA based Turing machine had been proposed by Charles Bennet [1]; he used imaginary enzymes to perform the transition rules. Shapiro's group [2] for the first time developed an autonomous 2-state molecular automaton made of DNA, oligonucleotides, *FokI* endonuclease and ligase. Tom Head [3] introduced splicing systems (H-system, the formal model of DNA recombination) and demonstrated the computation on a plasmid by restriction enzymes and ligase [4]. Beaver [5] proposed that a universal Turing machine can be simulated by substitutions on DNA. Adleman [6] used the molecular biology tools for the solution of hard combinatorial optimization problems, i.e., the Directed Hamiltonian Path Problem. Landweber and Kari [7] showed the rearrangements of DNA and RNA, such as scrambling or editing, and proved the potential for solving computational problems that occur in biological systems. Recently we suggested the role of lateral gene transfer in the biological evolution [8] and compared it with the computation by communication [9]. The idea of a 'universal endonuclease' was proposed by Szybalski [10] and further developed in the Schultz's laboratory [11, 12]. They conceived that the catalytic domain of an endonuclease may be linked to a double- or triplex-forming oligonucleotide to generate an enzyme with novel specificities. In other reports to achieve a sequence-specific cleavage of DNA by topoisomerase I the guide oligonucleotide was covalently linked with the ligand to Topo I [13, 14].

Our investigation of computing by guided nucleases was inspired by recent data concerning Argonaute proteins and siRNA/RISC complex [15], and by the paradigm shift from algorithms to computation by interaction [16]. We develop CPST computational model (*cut-paste-select-transpose*) that is close to Eilenberg P-systems [17]. Our logical construct is a good agreement with 'cut-and-paste' and 'copy-and-paste' model of bacterial evolution [18]. In order to achieve a molecular implementation we are currently designing the nuclease, which sequence-specificity can be guided by the labeled oligonucleotides or PNA. This could be seen as a 'molecular head' of the computing machine. We consider two possibilities: 1) reaction in the test-tube in a thermo-cycler, or 2) cellular computing after an installation of the transgene coding the nuclease into living cells. A scheme is proposed and analyzed, where initial results allow establishing perspectives and project on more complex problem.

2. An abstraction

Consider an evolving system – an abstract machine and an environment that is continuously changing creates input words for the machine to stimulate an adaptation of this device to the surrounding.

Description. The *Argo-machine* (***AM***) consists of a finite set of *agents* (*Argonauts*); each of these has a head, a finite tape and can be in different states, which we specify as the output states[1]. The tape is a nonempty string of symbols that may be linear or circular. The head scans the tape according to an input word w_i, and cuts the tape at recognized sites. The agent arbitrarily pastes the tape's fragments[2]. For each tape-configuration there is an appropriate output state of the agent that is checked by the environment. Agents take special 'accept' and 'reject' actions. An agent accepts, if its output state corresponds to the environment state; an agent will reject if less than two matches to the input word exist on the tape. ***AM*** can accept if at least one agent accepts, reject if all agents reject, or loop. If the environment has changed, then it delivers a transposition[3] and a new word w_{i+1}. ***AM*** looks for an agreement with the environment permanently (Fig.1).

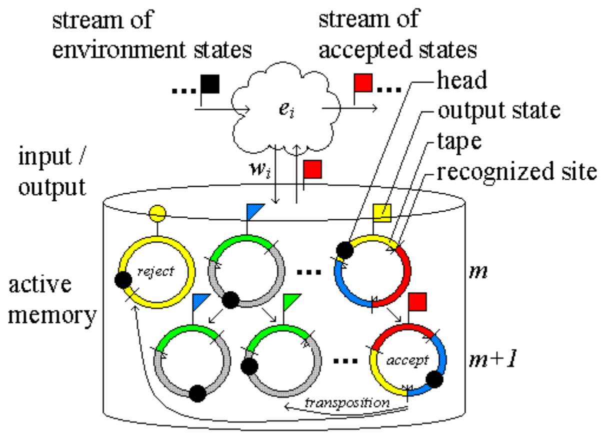

Fig.1 Schematic of *Argo-machine* (***AM***) with circular tapes.
The system operates on inputs and active memory, indirect uploads the memory and yields outputs (see text).

[1] The output state of the agent and the state of environment mean a string, a computable number (vector), a structure, e.g., the fractal [19] or a biological feature in applications. There is a mapping from the tape to the output state, from genotype to phenotype.
[2] The phrase "arbitrarily paste the tape" means to cut the tape into fragments and join them back together in an arbitrary order with or without the turn over. Fragments can not migrate between agents.
[3] The transposition means to make a copy of the tape from the accepted agent to other ones and join it in back-to-front.

In short, ***AM*** is a set of stochastic cut-paste agents, which act in parallel on their own tapes accordingly the instructions (input words), communicate with each other by transpositions of the tape and interact with the environment to compare the output states. Based on the comparison it accepts or runs in a loop to fitness to the environment. Each agent executes the same action, the *Argonaut's algorithm*.

Argonaut algorithm. $\mathscr{A}$ = "On word w:

1. Scan the tape to be sure that it contains at least two matches. If not, reject.
2. Cut at the matching sites and arbitrarily paste the tape's fragments.
3. Take the output state according the new tape.
4. Check it with the state of environment. If satisfy, accept; otherwise loop."

See Example 1 in the Appendix; here the states of the agent and an environment are represented by strings on the same alphabet, the tape is linear.

An Argo-machine is the 5-tuple system $\boldsymbol{AM} = (\Sigma, O, E, A, \zeta)$, where

1. Σ is a finite alphabet, I and T are nonempty sets, such that I is the 'oracle' language over Σ and T is the set of finite strings (tapes) over Σ;
2. $O = \{ct, pt, sl, tn\}$ is a finite set of operators on T, here ct, pt, sl, tn denotes cut, paste, select and transpose (copy) respectively;
3. E is an infinite set of environment states;
4. A is a finite set of agents, such as $\boldsymbol{A} = (T, S, \delta)$, where

- S is a large finite set of output states, including sets: B, F, R, where B is the subset of initial states, F is the subset of final states, R is the subset of reject states;
- δ is the transition function, which according to the algorithm $\mathscr{A}$ for each word $w_i \in I$ at the input of agent $a_j \in A$ after operations $\{ct, pt\}$ on the tape $t_j \in T$ assigns an output state $s_j \in S$ by the mapping f: $T \rightarrow S$. The state s_j that is compared with the current environment state $e_i \in E$. Formally this can be expressed as $\delta(w,t) = (ct, pt, s, e)$;

5. ζ is the super-transition function, which generates the transposition $\{tn\}$ of the tape $t_j \in T$ from the agent $a_j \in A$ in the state $s_j \in F$ to other agents $a^+ \in A$ after the selection event $\{sl\}$, assigns the new initial states $n^+ \in B$, and finally delivers the new word w_{i+1}, or formally $\zeta(e,s) = (sl, tn, n^+, w)$.

A computation scenario of the *Argo-machine* is the shuffling of tapes from an initial set T_0 until an accepted output state is reached, at which the device halts temporarily and is not evolving. We define it as an *adaptation*. In general, the computation never ends, because the environment changes permanently. The event of environment change is called a *catastrophe* if it causes a super-transition. A progression of adaptations and catastrophes is defined as *evolution*. The intermediate computation is defined to be the result produced by ***AM***, which is the fit to the current environmental state on input stream and attached at the left site of the output stream of accepted states (Fig.1). Because it is essentially driven by selection and input words we are using synonyms such as: 'oracle', 'guide', or 'generative' for the input words.

Transpositions can be seen as the messages from the winner agent to the other ones. The size of tapes of the agents may grow beyond a limit. Additionally, we consider the output states of agents, as the messages to an environment. The environment is not passive and has the attributes: it deals with a stream of states, it monitors the adaptive fitness of agents, it can also produce new evolution rules or new oracle words in this senses. In the best case, these words create the solutions, and transpositions increase the size of tapes and a complexity (see Example 2, Appendix), else the system runs in a competitive regime. On the other extreme, the system rejects these words. There are two main questions. *What oracle words are optimal for a creative combinatorial design? What are the rules to form the oracle language, which generates the library of solutions*?

An analysis. To demonstrate how the *Argo-machine* works, let us assume the word w_i, which leads to brakes at matching sites x on the tape t_j, then a random paste of the tape. The number of rearrangements (with the turn over) for the single circular tape is described by the combinatorial formula *r(x)*:

$$\begin{cases} r_0 = r_1 = 0, \\ r_x = 2^x * (x-1)!; x \geq 2 \end{cases} \tag{1}$$

where r_x is the number of rearrangements, x is the number of matching sites on the tape. To illustrate a combinatorial power of expression (1) we compare the discrete function *r(x)* with functions 2^x, e^x and *x!*, see Fig.2.

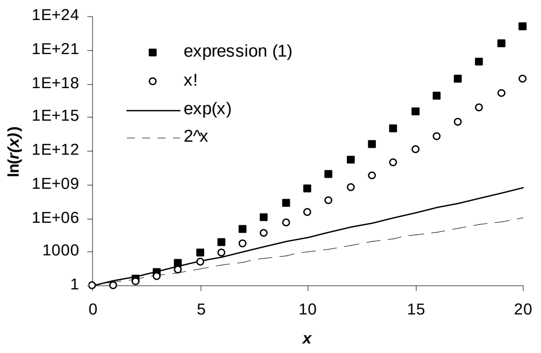

Fig.2 Combinatorial power of expression (1) versus 2^x, e^x, *x!*

Now consider the corresponding output state *s* that will be checked by the environment for each rearrangement *r*. According to the *Argonaut algorithm* $\mathscr{A}$, each agent a_j starts from the initial state s_j^0, runs on the oracle word w_i until the environment accepts at least one agent, then ***AM*** is stopped, Fig.3.

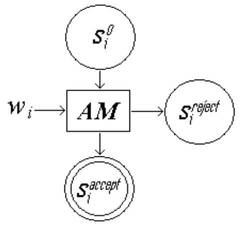

Fig.3 Evolving of *Argo-machine* from states s_j^0 to s_j^{accept} on word w_i ($s_j^0 \in B$, $s_j^{reject} \in R$, $s_j^{accept} \in F$, w_i is input word).

If the environment changes, then ***AM*** will require a new input. The catastrophe e_i generates the transposition tn_i, the winner agent concatenates a copy of its tape to another agents. After that the new initial states ${}_{i+1}s_j^0$ are formally assigned, the new word w_{i+1} starts up and the process runs ones more. If there is not any suitable word on input to satisfy the *Argonaut algorithm*, then all agents reject; now ***AM*** rejects and the process stops. We illustrate the evolution route on the example of winning agent's trace in Fig.4.

Computation is performed by a set of agents parallel in the nondeterministic manner. A computation power of this system depends on the number of agents and the number of output states for each agent. Note that it is impossible to run exponentially many agents simultaneously in real conditions; at least it needs a substitution of rejecting agents by winning ones. That is important the transpositions can propagate the local solutions and resolve some reject states. Transpositions enhance significantly the combinatorial-explosive search by increasing a dimension of the design space. This kind of distributed concurrent computation could permit a compositional design with the ruffle landscape. If a massive parallelism was achieved, then our model of computation would demonstrate a great computing power.

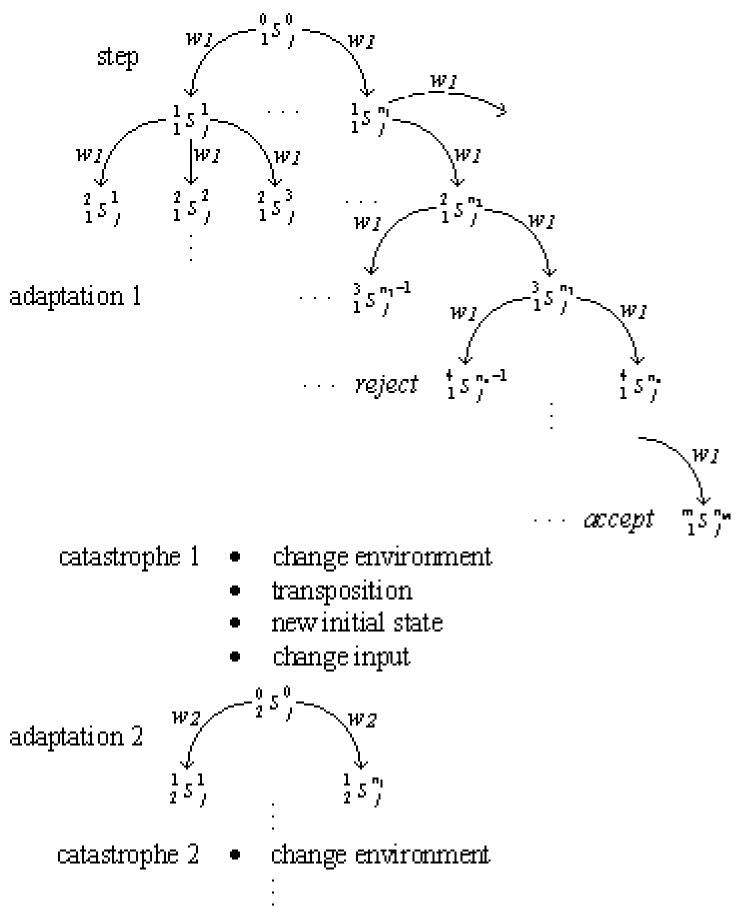

Fig.4 Nondeterministic computation by the agent *j*.
Here is ${}^m_i s^n_j$ - agent *j* in the state *n* on the step *m* before the catastrophe e_i, *i* - system's counter, w_i – oracle word.

3. Biological reality

We consider bio-molecules and living cells as suitable candidates to test the proposed *Argo-machine* in a real-world application. Besides, this molecular-scale machine may be a phenomenological model to understand the underlying biological processes.

Design. On the first step of investigation we created the fusion protein – IGNAF – comprising NucA non-specific endonuclease from cyanobacterium *Anabaena sp*. PCC 7120, and 4-4-20 scFv mouse single-chain antibody to fluorescein. Plasmids pBBInucA and pIG6Flullmh were used to construct pIGNucAFlu plasmid; pBBInucA was received from Alfred Pingoud (Giessen University), pIG6Flullmh was gifted by Andreas Pluckthun (Zurich University). *NucA* gene was cloned by PCR; a reverse primer encoded the sequence of tether. This cloned 780 bp fragment was inserted into pIG6Flullmh plasmid. As a result, IGNAF protein includes the ompA secretion signal, NucA domain, GSGGSGGSG peptide tether, variable light-chain (V_L) domain of scFv antibody, $(GGGGS)_6$ 30-mehr linker, variable heavy-chain (V_H) domain, and His-Tag. The C166G NucA mutant was obtained to avoid IGNAF dimer formation into the periplasma of *E.coli* cells. The additional mutations (R93A and W159A) were performed to decrease the nuclease activity and to avoid an inter-chain DNA cleavage. To increase the robustness of the enzyme two versions of protein are developed: (A) the monopod α-IGNAF is evolved by optimisation of NucA-domain via error prone PCR, shuffling and phage display, (B) the bipod β-IGNAF contains NucA split domain that is activated by self-assembling at the target DNA site. The split is located between β-sheet and α-helix in the T-P hinge region. Each part of NucA domain is linked to the own scFv antibody on N- and -C terminus accordingly (Fig.5). Thus, α-IGNAF has only a one 'leg' to contact to DNA by the specific oligonucleotide, although β-IGNAF has two 'legs' and theoretically is more robust.

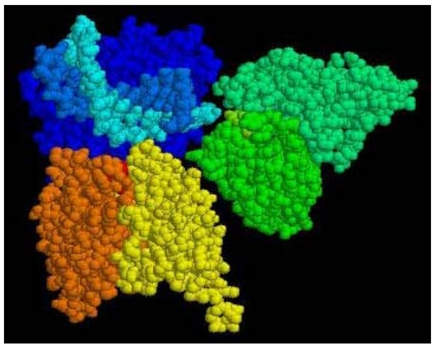

Fig.5 The model of β-version of IGNAF artificial nuclease on DNA.

Implementation. We suppose oligonucleotides or PNA labelled by fluorescein as *input-guide molecules*. IGNAF will bind to fluorescein and break DNA at the complementary sequence. We see two clearly distinguishable possibilities, the implementations *in vitro* (1) and *in vivo* (2).

1) The following alternatives *in vitro*:

a) the *catalytical approach* means that the nuclease is a catalytic with substrate turnover above the melting temperature T_m;

b) the *robust approach* will allow carrying out repeated hybridizations and cleavage reactions.

2) The required performances *in vivo*:

a) preinstallation of *ignaf*-transgene into living cells,

b) introduction of gene markers (*input-guide molecules*) in the cells,

c) activation of IGNAF nuclease at the target site.

Obviously IGNAF must specifically bind at a desired site, and then precisely cut the DNA chain in this region. Remarkably IGNAF is not an enzyme at the temperature less than T_m; it should be a genetic molecular device for applications *in vivo*, which acts only at the specific position. Smart IGNAF molecules have to bind at the target site, then switch on, next cleave DNA strand, and finally switch off. Thus under the control of guide molecules introduced into the system, computation will be performed in a loop, by the cleavage of the desired DNA pattern, and by a ligation to proceed the selected decision. Previous experiments demonstrated a limitation of specific cleavage by the 'monopod' guided nuclease [20]. To achieve the particular orientation of the nuclease on DNA the two different 'legs' would be more preferred. In this case the input comprising two half-words should be considered in the Argo-model. We do not discuss here a mechanism of transposition that could be implemented in the frame of the 'minimal cell' project. The compartmentalization of reactions is under the future investigations.

4. Conclusion

Computability and complexity are related to the development, adaptation, and evolution. Unfortunately, so far there is no computational definition of life; no minimal set of conditions needed for life to exist. This paper has been attempted to describe *adaptations*, *catastrophes* and *evolution* in computational terms. Following our research the requirements for a minimal life arise: How long should words and tapes be? What are the rules for the oracle words to effectively search by mapping in the design space? How many tapes and how much time does it require? Even under these constraints, our *Argo-machine* seems much like an oracle-choice evolved system, which performs an interactive 'single-instruction, multiple-data' computation in parallel. This concept combines *constructive*, *selective*, and *communicative* principles. This architecture could be applied to search the solutions of hard problems and to mimic the compositional evolution [21].

5. Acknowledgments

AK thanks Mikhail Kats for helpful discussion, Vladislav Erokhin for an implicit idea to use 'cut-paste' instead of 'read-write' operators, Alfred Pingoud and Andreas Pluckthun for DNAs, Albrecht Sippel for a creative lab environment, and mother who told him about Argonauts. AK is solely responsible for any mistake herein.

6. Appendix

Argo-machine, a computation in the winning branch.

Language notations:

```
~,<,(,... – strings, cut before open brackets;
# - boundary symbol
```

Example 1. Adaptation without transposition:

```
environment '<~~>', word '<'
 1. <~~>                     environment
 2. <                        word
 3. #~<~<~<~#                tape_tick_1
 4. #~<~~><~#                tape_tick_2
 5. <~~>                     accept
```

Example 2. Two adaptations with one transposition:

```
environment_1 '<~(~>', word_1 '<',
environment_2 '<~~~>', word_2 '('
 1. <~(~>                    environment_1
 2. <                        word_1
 3. #~(<~<~)<~#              tape_tick_1.1
 4. #~(<~(~><~#              tape_tick_1.2
 5. <~(~>                    accept_1
 6. <~~~>                    environment_2
 7. #~(<~<~)<~##~(<~(~><~#   transposition
 8. (                        word_2
 9. #~(<~<~)<~~(<~(~><~#     tape_tick_2.1
10. #~(<~<~)<~~~>)(~><~#     tape_tick_2.2
11. <~~~>                    accept_2
```

7. References

1. Bennett C. H. (1982). The thermodynamics of computation. *International Journal of Theoretical Physics*, **21**, 905–940.
2. Benenson Y., Paz-Elizur T., Adar R., Keinan E., Livneh Z., Shapiro E. (2001). Programmable and autonomous computing machine made of biomolecules. *Nature*, **22(414)**, 430-434.
3. Head T. (1987). Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, **49(6)**, 737-759.
4. Head T., Rozenberg G., Bladergroen R. S., Breek C. K., Lommerse P. H., Spaink H. P. (2000). Computing with DNA by operating on plasmids. *BioSystems*, **57**, 87-93.
5. Beaver D. (1996). A Universal Molecular Computer. *DNA Based Computers (Selected papers from Proc. of DIMACS Workshop on DNA Based Computers'95), DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, **27**, 29-36.
6. Adleman L. M. (1994). Molecular computation of solutions to combinatorial problems. *Science*, **266(5187)**, 1021-1024.
7. Landweber L. F., Kari L. (1999). The evolution of cellular computing: nature's solution to a computational problem. *Biosystems*, **52(1-3)**, 3-13.
8. Kuznetsov A. V., Kuznetsova I. V., Schit I. Yu. (2000). DNA interaction with rabbit sperm cells and its transfer into ova in vitro and in vivo. *Molecular Reproduction and Development*, **56(2)**, 292-297.

9. Kuznetsov A. V. (2004). The rules of sperm-mediated gene transfer. *Alife Mutants Hackingsession on Systems and Organisms (AMHSO), Rule 110 Winter Workshop, Bielefeld, Germany, 6-13 March.* Retrieved date, from http://www.rule110.org/amhso/results/gene-transfer.pdf
10. Szybalski W. (1985). Universal restriction endonucleases: designing novel cleavage specificities by combining adapter oligodeoxynucleotide and enzyme moieties. *Gene*, **40(2-3)**, 169-173.
11. Corey D. R., Schultz P. G. (1987). Generation of a hybrid sequence-specific single-stranded deoxyribonuclease. *Science*, **238(4832)**, 1401-1403.
12. Pei D., Corey D. R., Schultz P. G. (1990). Site-specific cleavage of duplex DNA by a semisynthetic nuclease via triple-helix formation. *Proc. Natl. Acad. Sci. USA,* **87**, 9858-9862.
13. Matteucci M., Lin K-Y., Huang T., Wagner R., Sternbach D. D., Mehrotra M., Besterman J. M. (1997). Sequence-specific targeting of duplex DNA using a camptothecin-triple helix forming oligonucleotide conjugate and topoisomerase I. *J. Am. Chem. Soc.*, **119**, 6939-6940.
14. Arimondo P. B., Bailly C., Boutorine A. S., Moreau P., Prudhomme M., Sun J. S., Garestier T., Helene C. (2001). Triple helix-forming oligonucleotides conjugated to indolocarbazole poisons direct topoisomerase I-mediated DNA cleavage to a specific site. *Bioconjug. Chem.,* **12(4)**, 501-509.
15. Parker J. S., Roe S. M., Barford D. (2005). Structural insights into mRNA recognition from a PIWI domain-siRNA guide complex. *Nature*, **434(7033)**, 663-666.
16. Wegner P, Goldin D. (2003). Computation Beyond Turing Machines. *Communications of the ACM*, **46(4)**, 100-102.
17. Gheorghe M., Holcombe M., Kefalas P. (2003). Eilenberg P systems: a bio-computational model. *Proc. First Balkan Conf. on Informatics, Thessaloniki, Greece*, 147-160.
18. Dawkins R. (2004). *The ancestor's tale: a pilgrimage to the dawn of evolution.* Boston: Houghton Mifflin.
19. Bentley P. J. (2004). Fractal Proteins. *Genetic Programming and Evolvable Machines Journal*, **5**, 71-101.
20. Corey D.R., Pei D., Schultz P.G. (1989) Generation of a catalytic sequence-specific hybrid DNase. *Biochemistry*, **28**, 8277-8286.
21. Watson R. A. (2006) Compositional Evolution: The Impact of Sex, Symbiosis, and Modularity on the Gradualist Framework of Evolution. (Vienna Series in Theoretical Biology) A Bradford Book. 324 p.