How to Compute with DNA*

Lila Kari[1], Mark Daley[1], Greg Gloor[2], Rani Siromoney[3], and Laura F. Landweber[4]

[1] Dept. of Computer Sci., Univ. of Western Ontario, London, ON N6A 5B7 Canada
lila@csd.uwo.ca, www.csd.uwo.ca/~lila
daley@csd.uwo.ca, www.csd.uwo.ca/~daley
[2] Dept. of Biochemistry, Univ. of Western Ontario, London ON N6A 5C1 Canada
ggloor@julian.uwo.ca, www.biochem.uwo.ca/fac/~gloor
[3] Dept. of Computer Sci., Madras Christian College, Madras 600 059 India
ranisiro@satyam.net.in
[4] Dept. of Ecology & Evolutionary Biology, Princeton Univ., NJ 08544-1003 USA
lfl@princeton.edu, www.princeton.edu/~lfl

Abstract. This paper addresses two main aspects of DNA computing research: DNA computing *in vitro* and *in vivo*. We first present a model of DNA computation developed in [5]: the circular insertion/deletion system. We review the result obtained in [5] stating that this system has the computational power of a Turing machine, and present the outcome of a molecular biology laboratory experiment from [5] that implements a small instance of such a system. This shows that rewriting systems of the circular insertion/deletion type are viable alternatives in DNA computation *in vitro*. In the second half of the paper we address DNA computing *in vivo* by presenting a model proposed in [17] and developed in [18] for the homologous recombinations that take place during gene rearrangement in ciliates. Such a model has universal computational power which indicates that, in principle, some unicellular organisms may have the capacity to perform any computation carried out by an electronic computer.

1 Introduction

Electronic computers are only the latest in a long chain of man's attempts to use the best technology available for doing computations. While it is true that their appearance, some 50 years ago, has revolutionized computing, computing does not start with electronic computers, and there is no reason why it should end with them. Indeed, even electronic computers have their limitations: there is only so much data they can store and their speed thresholds determined by physical laws will soon be reached. The latest attempt to break down these barriers is to replace, once more, the tools for doing computations: instead of electrical use biological ones.[13]

Research in this area was started by Leonard Adleman in 1994, [1], when he surprised the scientific community by using the tools of molecular biology to solve a hard computational problem. Adleman's experiment, solving an instance

C. Pandu Rangan, V. Raman, R. Ramanujam (Eds.): FSTTCS'99, LNCS 1738, pp. 269–282, 1999.

of the Directed Hamiltonian Path Problem solely by manipulating DNA strands, marked the first instance of a mathematical problem being solved by biological means. The experiment provoked an avalanche of computer science/molecular biology/biochemistry/physics research, while generating at the same time a multitude of open problems.[13]

The excitement DNA computing incited was mainly caused by its capability of massively parallel searches. This, in turn, showed its potential to yield tremendous advantages from the point of view of *speed, energy consumption* and *density of stored information*. For example, in Adleman's model, [2], the number of operations per second was up to 1.2×10^{18}. This is approximately 1,200,000 times faster than the fastest supercomputer. While existing supercomputers execute 10^9 operations per Joule, the energy efficiency of a DNA computer could be 2×10^{19} operations per Joule, that is, a DNA computer could be about 10^{10} times more energy efficient (see [1]). Finally, according to [1], storing information in molecules of DNA could allow for an information density of approximately 1 bit per cubic nanometer, while existing storage media store information at a density of approximately 1 bit per 10^{12} nm^3. As estimated in [3], a single DNA memory could hold more words than all the computer memories ever made.[12]

A few more words, as to why we should prefer biomolecules to electricity for doing computation: the short answer is that it seems more natural to do so. We could look at the electronic technology as just a technology that was in the right place at the right time; indeed, electricity hardly seems a suitable and intuitive means for storing information and for computations. For these purposes, nature prefers instead a medium consisting of biomolecules: DNA has been used for millions of years as storage for genetic information, while the functioning of living organisms requires computations. Such considerations seem to indicate that using biomolecules for computations is a promising new avenue, and that DNA computers might soon coexist with electronic computers.[13]

The research in the field has, from the beginning, had both experimental and theoretical aspects; for an overview of the research on DNA computing see [12]. This paper addresses both aspects. After introducing the basic notions about DNA in Section 2, in Section 3 we present a model of DNA computation developed in [5]: the circular insertion/deletion system. We show that this system has the computational power of a Turing machine and also present the results of a molecular biology laboratory experiment that implements a small instance of such a system. This shows that rewriting systems of the circular insertion/deletion type are viable alternatives in DNA computation *in vitro*. Section 4 introduces DNA computing *in vivo* by presenting a model proposed in [17], [18] for the homologous recombinations that take place during gene rearrangement in ciliates. We prove that such a model has universal computational power which indicates that, in principle, some unicellular organisms may have the capacity to perform any computation carried out by an electronic computer.

2 What is DNA?

DNA (deoxyribonucleic acid) is found in every cellular organism as the storage medium for genetic information. It is composed of units called nucleotides, distinguished by the chemical group, or base, attached to them. The four bases are *adenine, guanine, cytosine* and *thymine*, abbreviated as A, G, C, and T. (The names of the bases are also commonly used to refer to the nucleotides that contain them.) Single nucleotides are linked together end–to–end to form DNA strands. A short single-stranded polynucleotide chain, usually less than 30 nucleotides long, is called an *oligonucleotide* (or, shortly, oligo). The DNA sequence has a *polarity*: a sequence of DNA is distinct from its reverse. The two distinct ends of a DNA sequence are known under the name of the $5'$ end and the $3'$ end, respectively. Taken as pairs, the nucleotides A and T and the nucleotides C and G are said to be *complementary*. Two complementary single-stranded DNA sequences with opposite polarity will join together to form a double helix in a process called *base-pairing* or *annealing*. The reverse process – a double helix coming apart to yield its two constituent single strands – is called *melting*.[12]

A single strand of DNA can be likened to a string consisting of a combination of four different symbols, A, G, C, T. Mathematically, this means we have at our disposal a 4-letter alphabet $X = \{A, G, C, T\}$ to encode information. As concerning the operations that can be performed on DNA strands, the existing models of DNA computation are based on various combinations of the following primitive *bio-operations*, [12]:

– *Synthesizing* a desired polynomial-length strand.
– *Mixing:* pour the contents of two test-tubes into a third.
– *Annealing (hybridization):* bond together two single-stranded complementary DNA sequences by cooling the solution.
– *Melting (denaturation):* break apart a double-stranded DNA into its single-stranded components by heating the solution.
– *Amplifying (copying):* make copies of DNA strands by using the Polymerase Chain Reaction (PCR).
– *Separating* the strands by size using a technique called gel electrophoresis.
– *Extracting* those strands that contain a given pattern as a substring by using affinity purification.
– *Cutting* DNA double-strands at specific sites by using commercially available restriction enzymes.
– *Ligating:* paste DNA strands with compatible sticky ends by using DNA ligases.
– *Substituting:* substitute, insert or delete DNA sequences by using PCR site-specific oligonucleotide mutagenesis.
– *Detecting and Reading* a DNA sequence from a solution.

The bio-operations listed above and possibly others, will then be used to write "programs" which receive a tube containing DNA strands as input and return as output a set of tubes. A computation consists of a sequence of tubes containing DNA strands.

For further details of molecular biology terminology, the reader is referred to [12], [16].

3 How to Compute with DNA: Circular Insertions and Deletions

One of the aspects of the theoretical side of the DNA computing research comprises attempts to find a suitable model and to give it a mathematical foundation. This aspect is exemplified below by the *circular contextual insertion/deletion system*, [5] a formal language model of DNA computing. We mention the result obtained in [5] that the circular insertions/deletion systems are capable of universal computations. We also give the results of an experimental laboratory implementation of our model. This shows that rewriting systems of the circular insertion/deletion type are viable alternatives in DNA computation.

Insertions and deletions of small circular strands of DNA into/from long linear strands happen frequently in all types of cells and constitute also one of the methods used by some viri to infect a host. We describe here a generalization of insertions and deletions of words, [11], that aims to model these processes. (Note that circular DNA strings have been studied in the literature in the context of the splicing system model in [8], [9], [21], [24], [25].)[5]

In order to introduce our model, we first need some formal language definitions and notations. Throughout this paper, X represents an alphabet (a finite nonempty set), λ represents the empty word (the word containing 0 letters), $\bullet v$ represents a circular string v (a set containing every circular permutation of the linear string v). The length of a word v, denoted by $|v|$, is the number of occurrences of letters in v, counting repetitions. For a language L, by $\bullet L$ we denote the set of all words $\bullet v$ where $v \in L$. For further formal language definitions and notations the reader is referred to [22], [23].

In the style of [15], we define a *circular insertion/deletion system*, [5], as a tuple

$$ID^{\bullet} = (X, T, I^{\bullet}, D, \mathcal{A})$$

where X is an alphabet, $\text{card}(X) \geq 2$, $T \subseteq X$ is the terminal alphabet, $I^{\bullet} \subseteq (X^*)^5$ is the finite set of circular insertion rules, $D \subseteq (X^*)^3$ is the finite set of deletion rules, and $\mathcal{A} \subseteq X^+$ is a linear strand called the *axiom*.

A circular insertion rule in $I^{\bullet}$ is written as $(c_1, g_1, \bullet x, g_2, c_2)_I$ where (c_1, c_2) represents the *context of the insertion*, $\bullet x$ is the string to be inserted and (g_1, g_2) are the *guides*, i.e. the location where $\bullet x$ is cut.

Given the rule above, the guided contextual circular insertion of the circular string $\bullet x$ into a linear string u is performed as follows. The circular word $\bullet x$ is linearized by cutting it between g_1 and g_2 (provided $g_1 g_2$ occurs as a subword in x) and reading it clockwise starting from g_1 and ending at g_2. The resulting linear strand is then inserted into the linear word u, between c_1 and c_2. If $c_1 c_2$ does not occur as a subword in u no insertion can take place. An example of circular insertion is illustrated in Figure 1.

A deletion rule in D is written as $(c_1, x, c_2)_D$ where (c_1, c_2) represents the context of deletion and x is the string to be deleted.

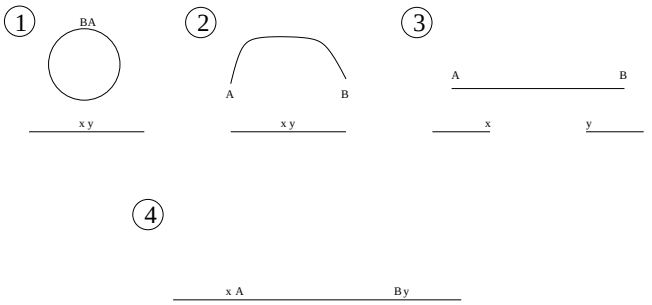

Fig. 1. Graphical representation of a circular insertion in the context (x, y), where the circular string is cut at the site (A, B). (From [5].)

Given the rule above, the linear contextual deletion of x from a linear word u accomplishes the excision of the linear strand x from u, provided x occurs in u flanked by c_1 on its left side and by c_2 on its right side.

If $u, v \in X^*$, we say that u derives v according to ID$^\bullet$ and we write $u \Rightarrow v$, [5], if v is obtained from u by either a guided contextual circular insertion or by a linear contextual deletion, i.e.,

– either $u = \alpha c_1 c_2 \beta$, $v = \alpha c_1 g_1 x' g_2 c_2 \beta$ and $I^\bullet$ contains the circular insertion rule $(c_1, g_1, \bullet x, g_2, c_2)_I$ where $g_1 x' g_2 \in \bullet x$, or

– $u = \alpha c_1 x c_2 \beta$, $v = \alpha c_1 c_2 \beta$ and D contains the linear deletion rule $(c_1, x, c_2)_D$.

A sequence of direct derivations

$$u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k, k \geq 0$$

is denoted by $u_1 \Rightarrow^* u_k$ and u_k is said to be derived from u_1.

The language $L(ID^\bullet)$, [5], accepted by the circular insertion/deletion system ID$^\bullet$ is defined as

$$L(ID^\bullet) = \{v \in T^* |\ v \Rightarrow^* \mathcal{A}, \mathcal{A} \text{ is the axiom } \}$$

Recall that, [23], a rewriting system $(S, X \cup \{\#\}, F)$ is called a *Turing machine* iff the following conditions are satisfied.

(i) S and $X \cup \{\#\}$ (with $\# \notin X$ and $X \neq \emptyset$) are two disjoint alphabets referred to as the *state* and *tape* alphabet.

(ii) Elements $s_0 \in S$, $\flat \in X$, and a subset $S_f \subseteq S$ are specified, namely, the *initial state*, the *blank symbol*, and the *final state set*. A subset $V_f \subseteq X$ is specified as the *final alphabet*.

(iii) The productions in F are of the forms

(1) $s_i a \ \rightarrow \ s_j b$ overprint
(2) $s_i ac \ \rightarrow \ a s_j c$ move right
(3) $s_i a\# \ \rightarrow \ a s_j \flat \#$ move right and extend workspace
(4) $c s_i a \ \rightarrow \ s_j ca$ move left
(5) $\# s_i a \ \rightarrow \ \# s_j \flat a$ move left and extend workspace

where $s_i, s_j \in S$ and $a, b, c \in X$. Furthermore, for each $s_i, s_j \in S$ and $a \in X$, F either contains no productions (2) and (3) (resp. (4) and (5)) or else contains both (2) and (3) (respectively (4), (5)) for every $c \in X$. For no $s_i \in S$ and $a \in X$, the word $s_i a$ is a subword of the left side in two productions of the forms (1), (3) and (5).

We say that a word sw, where $s \in S$ and $w \in (X \cup \{\#\})^*$ is *final* iff w does not begin with a letter a such that sa is a subword of the left side of some production in F. The language *accepted* by a Turing machine TM is defined by

$$L(TM) = \{w \in V_f^* |\ \#s_0 w\# \Rightarrow^* \#w_1 s_f w_2\# \text{ for some}$$

$$s_f \in S_f, w_1, w_2 \in X^* \text{ such that } s_f w_2\# \text{ is final}\}$$

where $\Rightarrow$ denotes derivation according to the rewriting rules (1) – (5) of the Turing machine. A language is *acceptable* by a Turing machine iff $L = L(TM)$ for some TM. It is to be noted that TM is *deterministic*: at each step of the rewriting process, at most one production is applicable.

The following result proved in [5] shows that the circular insertion/deletion systems defined above have the computational power of a Turing machine.

Theorem 1. *If a language is acceptable by a Turing machine TM, then there exists a circular insertion/deletion system $ID^\bullet$ accepting the same language.*

To test the empirical validity of our theoretical model, we implemented, [5], a small circular insertion/deletion system in the laboratory. The purpose of this implementation was to show that in vitro circular insertion is possible and not overwhelmingly difficult.

The following circular insertion/deletion system was chosen:

$$ID^\bullet = (X, T, I^\bullet, D, u)$$

where the alphabets are $T, X = \{A, C, G, T\}$, there are no deletion rules, i.e. $D = \emptyset$, the axiom u is a small DNA segment from the *Drosophila Melanogaster* genome and $I^\bullet = (G, G, \bullet v, TCGAC, TCGAC)$ where $\bullet v$ is a commercially available plasmid (circular strand). Note that A, C, G, T correspond to the four bases that occur in natural DNA, and that the sequence $G|TCGAC$ is the restriction (cut) site for the *Sal I* enzyme.[5]

To begin the experiment, we synthesized the linear axiom u in which we would then insert. This was accomplished by taking DNA from *Drosophila* (fruit fly) and performing PCR with the primers BC^+ and cd^-. The result was the amplification of a particular 682bp (basepair) linear sequence of DNA which became the axiom u of the circular insertion/deletion system. The 682bp linear strand was chosen to contain exactly one restriction site for the enzyme *Sal I*, corresponding to the context of insertion $(G, TCGAC)$. For the circular string $\bullet v$ to be inserted we chose pK18h, a commercially available plasmid having one restriction site for *Sal I*, corresponding to the guides $(G, TCGAC)$ in the insertion rule. [5]

After verifying that the PCR had worked correctly and we had indeed obtained the desired 682bp linear axiom u, we cut u with *Sal I*, cleaving it into two new linear strands denoted by L and R, i.e. $u = L|R$. The product was checked by gel electrophoresis to ensure the presence of bands corresponding to the sizes of L (188bp) and R (493bp), as seen from the first band in the gel of Figure 2. The plasmid $\bullet v$ was also cut and linearized in the same fashion resulting in the linear strand v. [5]

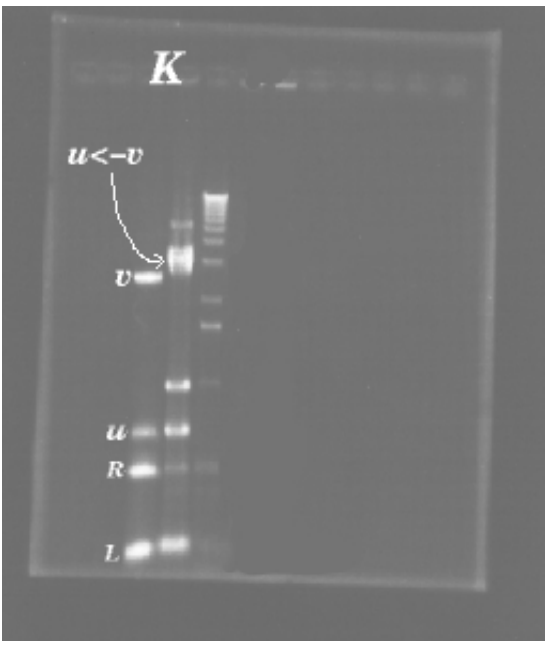

Fig. 2. The first vertical lane of this gel consists of bands corresponding to the unreacted linearized plasmid v, the linear strand u, and the two fragments of the cut linear strand (R, respectively L). The second vertical lane shows a band corresponding to the product obtained after reaction: the insertion of v into u, i.e., $u \leftarrow v$. The third lane contains a standard 1kb (kilobase) ladder used to measure the others. (From [5].)

At this point the linear strands L and R were combined with the linearized pK18h, i.e. v, and ligase was added to reconnect the strands of DNA. After allowing time for ligation, a gel was run to determine the products. The second band from the gel shown in Figure 2 indicates that in addition to the desired $L|$plasmid$|R$, we also obtain $R|R$, $L|R$, plasmid|plasmid and even plasmid | plasmid | plasmid. [5]

Note that the band corresponding to the approximate size of $L|\text{plasmid}|R$ can be seen as a smear. This could suggest the presence of $R|\text{plasmid}|R$ or of any other combination of two linear fragments and a plasmid which failed to separate clearly from one another due to the large size. Thus further analysis was required to ensure the presence of the desired product $L|\text{plasmid}|R$. [5]

In order to amplify the amount of DNA available at this point, the DNA was recircularized and introduced into *E. Coli* bacteria. (The complex details of this process are omitted here.)

Prior to sequencing, a restriction digest was performed on small amounts of product isolated from each of the several bacterial colonies. If the starting sample were a heterogeneous mixture of DNA molecules, each colony would yield a different product. Consequently, the restriction digest of DNA samples (each isolated from a particular colony) with enzymes *Sal I, Stu I* and *Xba* , resulted in bands indicating different size distributions. Of these, one sample corresponded to the size of $L|\text{plasmid}|R$ and the identity of the product was confirmed by sequencing. [5]

This experiment demonstrates that it is possible to insert a plasmid into a linear strand in vitro, implementing thus a circular insertion/deletion system. Future experimental work would ideally include a much larger system to test the scalability of this approach. [5]

4 How do Cells Compute?

The previous section presented one of the existing models of DNA computation and presented a toy experimental laboratory implementation. Despite the progress achieved in this direction of research, the main obstacles to creating an *in vitro DNA computer* still remain to be overcome. These obstacles are mainly practical, arising from difficulties in coping with the error rates of bio-operations, and with scaling up the existing systems. However, note that similar issues of actively adjusting the concentrations of reactions and fault detection and tolerance are all addressed by biological systems in nature: cells. This leads to another direction of research, *DNA computing in vivo*, which addresses the computational capabilities of cellular organisms.

Here we describe a model proposed in [17] and developed in [18] for the homologous recombinations that take place during gene rearrangement in ciliates and prove that such a model has the computational power of a Turing machine. This indicates that, in principle, these unicellular organisms may have the capacity to, perform at least any computation carried out by an electronic computer.

Ciliates are a diverse group of 8000 or more unicellular eukaryotes (nucleated cells) named for their wisp-like covering of cilia. They possess two types of nuclei: an active *macronucleus* (soma) and a functionally inert *micronucleus* (germline) which contributes only to sexual reproduction. The somatically active macronucleus forms from the germline micronucleus after sexual reproduction, during the course of development. The genomic copies of some protein-coding genes in the

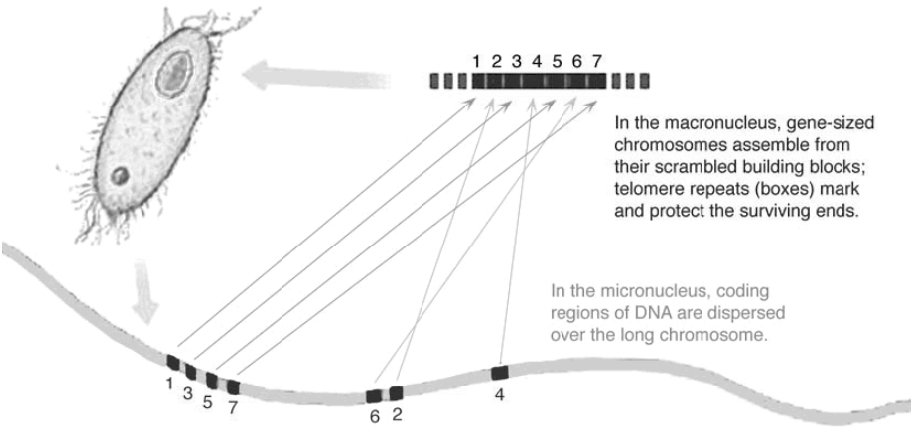

Fig. 3. Overview of gene unscrambling. Dispersed coding MDSs 1-7 reassemble during macronuclear development to form the functional gene copy (top), complete with telomere addition to mark and protect both ends of the gene. (From [17].)

micronucleus of hypotrichous ciliates are obscured by the presence of intervening non-protein-coding DNA sequence elements (internally eliminated sequences, or *IES*s). These must be removed before the assembly of a functional copy of the gene in the somatic macronucleus. Furthermore, the protein-coding DNA segments (macronuclear destined sequences, or *MDS*s) in species of *Oxytricha* and *Stylonychia* are sometimes present in a permuted order relative to their final position in the macronuclear copy. For example, in *O. nova*, the micronuclear copy of three genes (Actin I, α-telomere binding protein, and DNA polymerase α) must be reordered and intervening DNA sequences removed in order to construct functional macronuclear genes. Most impressively, the gene encoding DNA polymerase α (DNA pol α) in *O. trifallax* is apparently scrambled in 50 or more pieces in its germline nucleus [10]. Destined to unscramble its micronuclear genes by putting the pieces together again, *O. trifallax* routinely solves a potentially complicated computational problem when rewriting its genomic sequences to form the macronuclear copies. [18]

This process of unscrambling bears a remarkable resemblance to the DNA algorithm Adleman [1] used to solve a seven-city instance of the Directed Hamiltonian Path Problem. The developing ciliate macronuclear "computer" (Figure 3) apparently relies on the information contained in short direct repeat sequences to act as minimal guides in a series of homologous recombination events. These guide-sequences act as splints, and the process of recombination results in linking the protein-encoding segments (MDSs, or "cities") that belong next to each other in the final protein coding sequence. As such, the unscrambling of sequences that

encode DNA polymerase α accomplishes an astounding feat of cellular computation. Other structural components of the ciliate chromatin presumably play a significant role, but the exact details of the mechanism are still unknown.[18]

In this section we define the notion of a *guided recombination system*, [18]), that models the process taking place during gene rearrangement, and prove that such systems have the computational power of a Turing machine, the most widely used theoretical model of electronic computers.

The following strand operations generalize the intra- and intermolecular recombinations defined in [17] and illustrated in Figure 4 by assuming that homologous recombination is influenced by the presence of certain contexts, i.e., either the presence of an IES or an MDS flanking a junction sequence. The observed dependence on the old macronuclear sequence for correct IES removal in *Paramecium* suggests that this is the case ([19]). This restriction captures the fact that the guide sequences do not contain all the information for accurate splicing during gene unscrambling. [18]

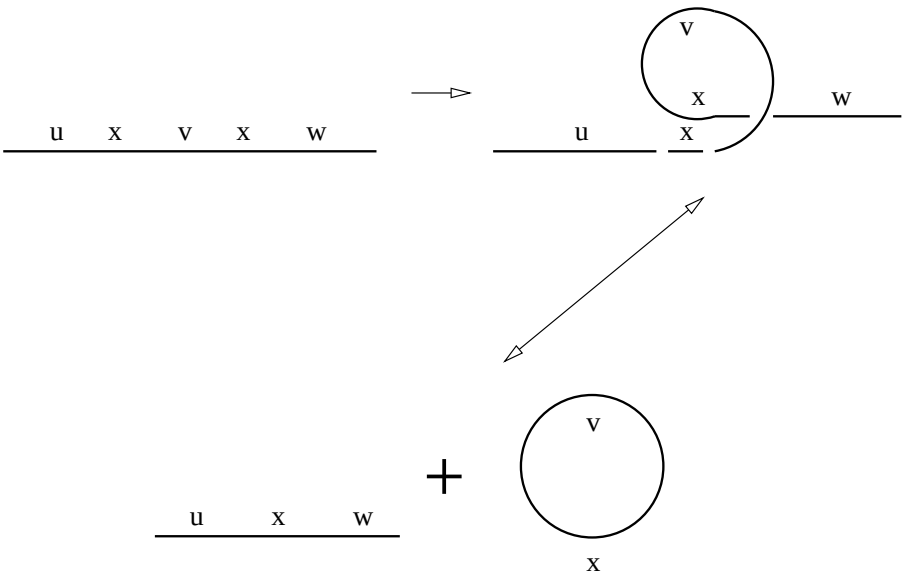

Fig. 4. Intra- and intermolecular recombinations using repeats x. During intramolecular recombination, after x finds its second occurrence in $uxvxw$, the molecule undergoes a strand exchange in x that leads to the formation of two new molecules: a linear DNA molecule uxw and a circular one $\bullet vx$. The reverse operation is intermolecular recombination. (From [14].)

Using an approach developed in [15] we use contexts to restrict the use of recombinations. A *splicing scheme*, [7], [8] is a pair $(X, \sim)$ where X is the alphabet

and $\sim$, the pairing relation of the scheme, is a binary relation between triplets of nonempty words satisfying the following condition: If $(p, x, q) \sim (p', y, q')$ then $x = y$. In the splicing scheme $(X, \sim)$ pairs $(p, x, q) \sim (p', x, q')$ now define the contexts necessary for a recombination between the repeats x. Then we define *contextual intramolecular recombination*, [18] as

$$\{uxwxv\} \Rightarrow \{uxv, \bullet wx\}, \text{ where } u = u'p, w = qw' = w''p', v = q'v'.$$

This constrains intramolecular recombination within $uxwxv$ to occur only if the restrictions of the splicing scheme concerning x are fulfilled, i.e., the first occurrence of x is preceded by p and followed by q and its second occurrence is preceded by p' and followed by q'. (See Figure 4.)

Similarly, if $(p, x, q) \sim (p', x, q')$, then we define *contextual intermolecular recombination*, [18]), as

$$\{uxv, \bullet wx\} \Rightarrow \{uxwxv\} \text{ where } u = u'p, v = qv', w = w'p' = q'w''.$$

Informally, intermolecular recombination between the linear strand uxv and the circular strand $\bullet wx$ may take place only if the occurrence of x in the linear strand is flanked by p and q and its occurrence in the circular strand is flanked by p' and q'. Note that sequences p, x, q, p', q' are nonempty, and that both contextual intra- and intermolecular recombinations are reversible by introducing pairs $(p, x, q') \sim (p', x, q)$ in $\sim$. (See Figure 4.)

The above operations resemble the "splicing operation" introduced by Head in [7] and "circular splicing" ([8], [24], [21]). [20], [4] and subsequently [25] showed that these models have the computational power of a universal Turing machine. (See [22] for a review.)

The operations defined in [17] are particular cases of guided recombination, where all the contexts are empty, i.e, $(\lambda, x, \lambda) \sim (\lambda, x, \lambda)$ for all $x \in X^+$. This corresponds to the case where recombination may occur between every repeat sequence, regardless of the contexts. These unguided (context-free) recombinations are computationally not very powerful: we can prove that they can only generate regular languages. [18]

If we use the classical notion of a set, we can assume that the strings entering a recombination are available for multiple operations. Similarly, there would be no restriction on the number of copies of each strand produced by recombination. However, we can also assume some strings are only available in a limited number of copies. Mathematically this translates into using *multisets*, where one keeps track of the number of copies of a string at each moment. In the style of [6], if $\mathbf{N}$ is the set of natural numbers, a multiset of X^* is a mapping $M : X^* \longrightarrow \mathbf{N} \cup \{\infty\}$, where, for a word $w \in X^*$, $M(w)$ represents the number of occurrences of w. Here, $M(w) = \infty$ means that there are unboundedly many copies of the string w. The set $\text{supp}(M) = \{w \in X^* | \ M(w) \neq 0\}$, the *support of* M, consists of the strings that are present at least once in the multiset M. [12]

We now define a *guided recombination system* that captures the series of dispersed homologous recombination events that take place during scrambled gene rearrangements in ciliates.

Definition. ([18]) *A guided recombination system is a triple* $R = (X, \sim, \mathcal{A})$ *where* $(X, \sim)$ *is a splicing scheme, and* $\mathcal{A} \in X^+$ *is a linear string called the axiom.*

A guided recombination system R defines a *derivation relation*, [18], that produces a new multiset from a given multiset of linear and circular strands, as follows. Starting from a "collection" (multiset) of strings with a certain number of available copies of each string, the next multiset is *derived from* the first one by an intra- or inter-molecular recombination between existing strings. The strands participating in the recombination are "consumed" (their multiplicity decreases by 1) whereas the products of the recombination are added to the multiset (their multiplicity increases by 1).

For two multisets S and S' in $X^* \cup X^\bullet$, we say that S derives S' and we write $S \Rightarrow_R S'$, iff one of the following two cases hold, [18]:

(1) there exist $\alpha \in \text{supp}(S)$, $\beta, \bullet\gamma \in \text{supp}(S')$ such that
– $\{\alpha\} \Rightarrow \{\beta, \bullet\gamma\}$ according to an intramolecular recombination step in R,
– $S'(\alpha) = S(\alpha) - 1$, $S'(\beta) = S(\beta) + 1$, $S'(\bullet\gamma) = S(\bullet\gamma) + 1$;

(2) there exist $\alpha', \bullet\beta' \in \text{supp}(S)$, $\gamma' \in \text{supp}(S')$ such that
– $\{\alpha', \bullet\beta'\} \Rightarrow \{\gamma'\}$ according to an intermolecular recombination step in R,
– $S'(\alpha') = S(\alpha') - 1$, $S'(\bullet\beta') = S(\bullet\beta') - 1$, $S'(\gamma') = S(\gamma') + 1$.

Those strands which, by repeated recombinations with initial and intermediate strands eventually produce the axiom, form the language *accepted* by the guided recombination system. Formally, [18],

$$L_a^k(R) = \{w \in X^* |\ \ \{w\} \Rightarrow_R^* S \text{ and } \mathcal{A} \in \ \text{supp}(S)\},$$

where the the multiplicity of w equals k. Note that $L_a^k(R) \subseteq L_a^{k+1}(R)$ for any $k \geq 1$.

Theorem.([18])*Let* L *be a language over* T^* *accepted by a Turing machine* $TM = (S, X \cup \{\#\}, P)$ *as above. Then there exist an alphabet* X'*, a sequence* $\pi \in X'^*$*, depending on* L*, and a recombination system* R *such that a word* w *over* T^* *is in* L *if and only if* $\#^6 s_0 w \#^6 \pi$ *belongs to* $L_a^k(R)$ *for some* $k \geq 1$.

The preceding theorem implies that if a word $w \in T^*$ is in $L(TM)$, then $\#^6 s_0 w \#^6 \pi$ belongs to $L_a^k(R)$ for some k and therefore it belongs to $L_a^i(R)$ for any $i \geq k$. This means that, in order to simulate a computation of the Turing machine on w, any sufficiently large number of copies of the initial strand will do. The assumption that sufficiently many copies of the input strand are present at the beginning of the computation is in accordance with the fact that there are multiple copies of each strand available during the (polytene chromosome) stage where unscrambling occurs. Note that the preceding result is valid even if we allow interactions between circular strands or within a circular strand, particular cases of which have been formally defined in [17].[18]

The proof that a guided recombination system can simulate the computation of a Turing machine suggests that the micronuclear gene, present in multiple

copies, consists of a sequence encoding the input data, combined with a sequence encoding a program, i.e., a list of encoded computation instructions. The "computation instructions" can be excised from the micronuclear gene and become circular "rules" that can recombine with the data. The process continues then by multiple intermolecular recombination steps involving the linear strand and circular "rules", as well as intramolecular recombinations within the linear strand itself. The resulting linear strand, which is the functional macronuclear copy of the gene, can then be viewed as the output of the computation performed on the input data following the computation instructions excised as circular strands. [18]

The last step, telomere addition and the excision of the strands between the telomere addition sites, can easily be added to our model as a final step consisting of the deletion of all the markers, rule delimiters and remaining rules from the output of the computation. This would result in a strand that contains only the output of the computation (macronuclear copy of the gene) flanked by end markers (telomere repeats). This also provides a new interpretation for some of the vast quantity of non-encoding DNA found in micronuclear genes.[14]

In conclusion, in this section we presented a model proposed in [17] for the process of gene unscrambling in hypotrichous ciliates. While the model is consistent with our limited knowledge of this biological process, it needs to be rigorously tested using molecular genetics. We have shown, however, that the model is capable of universal computation. This both hints at future avenues for exploring biological computation and opens our eyes to the range of complex behaviors that may be possible in ciliates, and potentially available to other evolving genetic systems. [18]

References

1. L.Adleman. Molecular computation of solutions to combinatorial problems. *Science* v.266, Nov.1994, 1021–1024. 269, 270, 270, 277
2. L.Adleman. On constructing a molecular computer. *1st DIMACS workshop on DNA based computers*, Princeton, 1995. In *DIMACS series*, vol.27 (1996), 1–21. 270
3. E.Baum. Building an associative memory vastly larger than the brain. *Science*, vol.268, April 1995, 583–585. 270
4. E. Csuhaj-Varju, R.Freund, L.Kari, and G. Păun. DNA computing based on splicing: universality results. In Hunter, L. and T. Klein (editors). *Proceedings of 1st Pacific Symposium on Biocomputing*. World Scientific Publ., Singapore, 1996, 179-190. 279
5. M.Daley, L.Kari, G.Gloor, R.Siromoney. Circular contextual insertions/deletions with applications to biomolecular computation. Proceedings of *String Processing and Information REtrieval '99*, Mexico, IEEE CS Press, 1999, in press. 269, 269, 269, 270, 272, 272, 272, 273, 273, 273, 274, 274, 275
6. S.Eilenberg. *Automata, Languages and Machines*. Academic Press, New York, 1984. 279
7. T.Head. Formal language theory and DNA: an analysis of the generative capacity of recombinant behaviors. *Bulletin of Mathematical Biology*, 49(1987), 737–759. 278, 279

8. T. Head. Splicing schemes and DNA. *Lindenmayer systems*, G.Rozenberg and A. Salomaa eds., Springer Verlag, Berlin, 1991, 371–383. 272, 278, 279
9. T.Head, G.Păun, D.Pixton. Language theory and genetics. Generative mechanisms suggested by DNA recombination. In *Handbook of Formal Languages* (G.Rozenberg, A.Salomaa eds.), Springer Verlag, 1996. 272
10. D.C.Hoffman, and D.M. Prescott. Evolution of internal eliminated segments and scrambling in the micronuclear gene encoding DNA polymerase α in two *Oxytricha* species. *Nucl. Acids Res.* 25(1997), 1883-1889. 277
11. L.Kari. On insertions and deletions in formal languages. *Ph.D. thesis*, University of Turku, Finland, 1991. 272
12. L.Kari. DNA computing: arrival of biological mathematics. *The Mathematical Intelligencer*, vol.19, nr.2, Spring 1997, 9–22. 270, 271, 271
13. L.Kari. From Micro-Soft to Bio-Soft: Computing with DNA. Proceedings of BCEC'97 (Bio-Computing and Emergent Computation) Skovde, Sweden, World Scientific Publishing Co., 146–164.
14. L.Kari, L.F.Landweber. Computational power of gene rearrangement. Proceedings of *DNA Based Computers V*, E.Winfree, D.Gifford eds., MIT, Boston, June 1999, 203-213. 278
15. L.Kari, G.Thierrin. Contextual insertions/deletions and computability. *Information and Computation*, 131, no.1 (1996), 47–61. 272, 278
16. J.Kendrew et al., eds. *The Encyclopedia of Molecular Biology*, Blackwell Science, Oxford, 1994. 271
17. L.F.Landweber, L.Kari. The evolution of cellular computing: nature's solution to a computational problem. Proceedings of 4th DIMACS meeting on DNA based computers, Philadephia, 1998, 3-15. 269, 270, 276, 277, 278, 279, 280, 281
18. L.F.Landweber, L.Kari. Universal molecular computation in ciliates. In *Evolution as Computation*, L.F.Landweber, E,Winfree, Eds., Springer Verlag, 1999. 269, 270, 276, 278, 279, 279, 280, 280, 280, 280, 280
19. E.Meyer,and S.Duharcourt. Epigenetic Programming of Developmental Genome Rearrangements in Ciliates. *Cell* (1996) 87, 9-12. 278
20. G.Păun. On the power of the splicing operation. *Int. J. Comp. Math* 59(1995), 27-35. 279
21. D.Pixton. Linear and circular splicing systems. Proceedings of the *First International Symposium on Intelligence in Neural and Biological Systems* , IEEE Computer Society Press, Los Alamos, 1995, 181–188. 272, 279
22. G.Rozenberg, and A.Salomaa eds. *Handbook of Formal Languages*, Springer Verlag, Berlin, 1997. 272, 279
23. A.Salomaa. *Formal Languages*. Academic Press, New York, 1973. 272, 273
24. R. Siromoney, K.G. Subramanian and Dare Rajkumar, Circular DNA and splicing systems. In *Parallel Image Analysis. Lecture Notes in Computer Science* 654, Springer Verlag, Berlin, 1992, 260–273. 272, 279
25. T. Yokomori, S. Kobayashi and C. Ferretti. Circular splicing systems and DNA computability Proc. of *IEEE International Conference on Evolutionary Computation'97*, 1997, 219–224. 272, 279