10

Biocomputation in Ciliates

Lila Kari and Laura F. Landweber

Ciliates are unicellular protists that may have arisen more than a billion years ago. They have since diverged into thousands of species, many uncharacterized, the genetic divergence among ciliates being at least as deep as that between plants and animals [17]. Despite their diversity, ciliates are united by two common features; the presence of short threads called cilia on their surface, whose rhythmic beating causes movement and is also useful for food capture, and the presence of two types of nuclei. The *macronucleus* contains DNA encoding functional copies of all the genes that regulate vegetative growth and cell proliferation. The *micronucleus* contains encrypted versions of the macronuclear DNA, is mostly functionally inert, and is only used for sexual exchange of DNA. In this chapter we study the decryption of the macronuclear DNA from a computational perspective.

INTRODUCTION

When two cells mate, they exchange micronuclear information. After they separate, the old micronuclei and macronuclei degenerate, while the newly formed micronuclei develop into new macronuclei over hours or days, depending on the species. Few ciliates have so far been studied at the level of molecular genetics: *Tetrahymena* and *Paramecium* representing the *Oligohymenophorans* and *Oxytricha* (recently renamed *Sterkiella*), and *Stylonichia* and *Euplotes* representing *Spirotrichs*. The DNA molecule in each of the approximately 120 chromosomes in the micronucleus contains on average approximately 10^7 basepairs (bp) in *Oxytricha* species and approximately 18×10^6 bp in *Stylonichia lemnae* [17]. The size of the DNA molecules in the macronucleus is, in contrast, very small. In various *Oxytricha* species and *S. lemnae*, macronuclear DNA molecules range in size from 400 to 15,000 bp with most molecules in the 1000–8000 bp range [18].

Macronuclear DNA sequences are derived from the micronuclear sequences through a series of DNA rearrangements as follows. The segments that together constitute a macronuclear sequence (*macronuclear destined sequences* or MDSs) are present as sub-sequences in the micronuclear DNA. However, in the micronuclear DNA, MDSs are interspersed with long DNA sequences (*internal eliminated sequences* or IESs) that are excised in the micronucleus to macronucleus differentiation. (Note that excision of IESs from micronuclear DNA is distinct from excision of introns, which occurs at the RNA level after transcription of macronuclear DNA.) IESs and intergenic DNA represent large regions of the micronuclear DNA. In *Oxytricha* species, only 4% of the micronuclear DNA represents macronuclear destined sequences, while in *S. lemnae* that proportion is still smaller, 2% [18].

In addition, in some spirotrich micronuclear genes, the MDSs are present in a permuted order in the micronuclear DNA, relative to the "correct" macronuclear order [18], as shown in Figure 10.1. For example, the micronuclear actin I gene in *Oxytricha nova* consists of nine MDSs separated by eight IESs. The order of MDSs in the micronucleus is 3-4-6-5-7-9-2-1-8 [19].

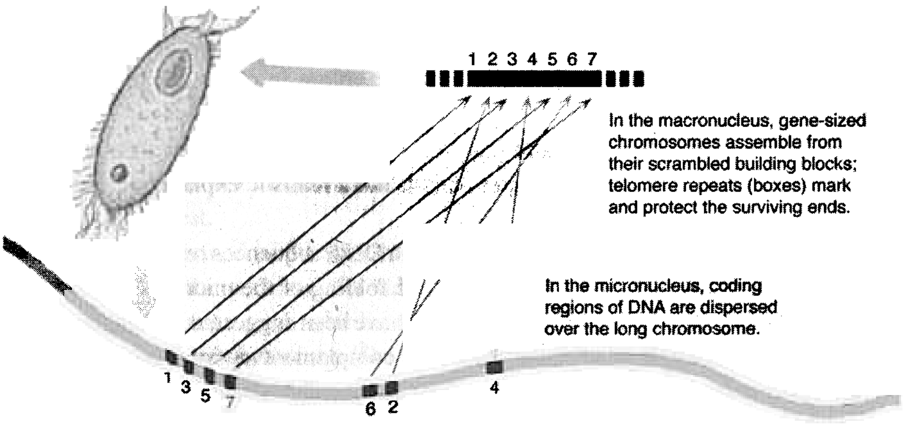
1 2 3 4 5 6 7
In the macronucleus, gene-sized chromosomes assemble from their scrambled building blocks; telomere repeats (boxes) mark and protect the surviving ends.
In the micronucleus, coding regions of DNA are dispersed over the long chromosome.
1 3 5 7
6 2
4

Figure 10.1 Overview of gene unscrambling. Dispersed coding macronuclear destined segments 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 Landweber and Kari [8].

The gene encoding the α telomere binding protein in *Oxytricha nova* is present in the micronucleus in the permuted order 1-3-5-7-9-11-2-4-6-8-10-12-13-14 [12]. The gene encoding DNA polymerase α in *S. lemnae* is apparently broken into more than 48 MDSs scrambled in an odd/even order [10], with 14 MDSs inverted on the opposite strand of another 24 MDSs, 9 additional MDSs on a separate locus, and 1 MDS not present on either locus. MDSs have to be reordered and intervening IESs eliminated to construct the functional macronuclear gene.

The process of unscrambling the micronuclear DNA into the macronuclear genes is ostensibly aided by the presence of short pointer sequences present at the junctions between MDSs and IESs. More precisely, at the junction between the nth MDS and the adjacent IES following, it there is a sequence (of 9–13 bp in actin I and 6–19 bp in α telomere binding protein) that is repeated somewhere else in the gene—namely, at the junction between the $(n+1)^{st}$ MDS and the adjacent IES preceding it. After aligning two pointers, homologous recombination between them would then join MDSs n and $(n+1)$ in the correct order and eliminate one copy of the pointer.

This process can be viewed as a computation solved during gene unscrambling by homologous DNA recombinations. If we assume that the cell's biochemistry can identify those DNA segments that represent pointers, and if the pointer pairs were unique in the micronuclear DNA, then one could argue that the ciliate is solving the computational problem of sorting a permuted sequence in the correct order. However, the computational problem facing the cell is much more complex given that some pointer sequences occur more than 13 times in a single gene (e.g., DNA polymerase α in *S. lemnae*). Taking into account the multiplicities of each pointer (the raw number of occurrences of the sequence representing the pointer in the micronuclear sequence), the number of combinations the cell would need to explore in order to arrive at the correct solution could be greater than 14 trillion for DNA polymerase α in *S. lemnae* [2]. Clearly, even assuming a priori knowledge of the pointer sequences, blind searching of matching pointer pairs is not a realistic explanation of gene unscrambling.

Other factors such as knowledge of which DNA sequences represent MDSs, IESs, and pointers, together with geometric folding of the micronuclear DNA that brings corresponding pointers together, have been suggested as mechanisms of gene unscrambling [17]. Alternative or complementary information guiding unscrambling may be the presence of contexts that flank correct matching pointer pairs and that might be responsible for solving this seemingly difficult computational problem [17, 18]. The details of the gene rearrangement process are still elusive. In the following sections we explore formal models for the homologous recombinations that lead to gene unscrambling in ciliates and investigate their computational power.

CLASSIC COMPUTATIONAL MODELS

In order to study the computational power of the bio-operations underlying gene rearrangement in ciliates, we compare them with the existing models of computation. In this section, we introduce two classic models of computation, the finite automaton and the Turing machine. The finite automaton has very restricted computational power, but the Turing machine models the computing capability of a general-purpose computer.

The finite automaton is a mathematical model of a system with a finite number of internal states and with discrete inputs and outputs. The behavior of the system is dictated by a finite number of rules that, given a state of the system and an input, dictate what the next state of the system will be.

To formalize the notions of finite automaton and Turing machine, let us first introduce some notations. An alphabet Σ is a finite, nonempty set of symbols or letters. A sequence of letters from Σ is called a string (word) over Σ. (If $\Sigma = \{A, C, G, T\}$, a word over Σ can be interpreted as a linear DNA strand.) The words are denoted by lowercase letters such as u, v, α_i, x_{ij}. A word with 0 letters in it is called an empty word and is denoted by λ. The set of all possible words consisting of letters from Σ is denoted by Σ^*, and the set of all nonempty words by Σ^+.

A finite automaton is a construct $A = (S, \Sigma, P, s_0, F)$, where Σ is the input alphabet, S is a finite set of states, $s_0 \in S$ is a designated start state, $F \subseteq S$ is the set of final states, and $P \subseteq S \times \Sigma \longrightarrow S$ is a set of transition rules. A transition rule $sa \longrightarrow s', s, s' \in S, a \in \Sigma$ says that, if the automaton is in state s and reads the input letter a, then it changes its state to the new state s' and continues scanning the input word. The language accepted by the automaton A is defined as:

$$L(A) = \{w \in \Sigma^* \mid s_0 w \Longrightarrow^* s_f, s_f \in F\};$$

in other words, the set of all input words that can take the automaton from the initial state to a final state by successive applications (denoted by $\Longrightarrow^*$) of the transition rules in P. In the hierarchy of computational models, finite automata are the weakest.

At the other end of the spectrum of computational power is the Turing machine (TM), the accepted formal model of what we call computation. In a Turing machine, a read/write head scans an infinite tape composed of discrete squares, one square at a time. The read/write head communicates with a control mechanism under which it can read the symbol in the current square or replace it by another. The read/write head is also able to move on the tape, one square at a time, to the right and to the left. The set of words that make a Turing machine finally halt is considered its language.

Formally [20], a rewriting system $\mathrm{TM} = (S, \Sigma \cup \{\#\}, P)$ is called a Turing machine if and only if (iff):

(1) S and $\Sigma \cup \{\#\}$ (with $\# \notin \Sigma$ and $\Sigma \neq \emptyset$) are two disjoint alphabets referred to as the *state* and the *tape* alphabets.
(2) Elements s_0 and s_f of S and B of Σ are the *initial* and *final* state, and the *blank symbol*, respectively. Also, a subset T of Σ is specified and referred to as the *terminal* alphabet. It is assumed that T is not empty.
(3) The productions (rewriting rules) of P are of the forms

(i) $s_i a \longrightarrow s_j b$ (overprint)
(ii) $s_i ac \longrightarrow as_j c$ (move right)
(iii) $s_i a\# \longrightarrow as_j B\#$ (move right and extend workspace)
(iv) $cs_i a \longrightarrow s_j ca$ (move left)
(v) $\#s_i a \longrightarrow \#s_j Ba$ (move left and extend workspace)
(vi) $s_f\, a \longrightarrow s_f$
(vii) $a\, s_f \longrightarrow s_f$

where s_i and s_j are in S, $s_i \neq s_f$, $s_j \neq s_f$, and a, b, c are in Σ. For each pair (s_i, a), where s_i and a are in the appropriate ranges, P either contains no productions (ii) and (iii) (respectively, iv and v) or else contains both (iii) and (ii) for every c (respectively contains both (v) and (iv) for every c). There is no pair (s_i, a) such that the word $s_i a$ is a subword of the left side in two productions of the forms i, iii, v.

A configuration of the TM is of the form $\#w_1 s_i w_2\#$, where $w_1 w_2$ represents the contents of the tape, #s are the boundary markers, and the position of the state symbol s_i indicates the position of the read/write head on the tape: if s_i is positioned at the left of a letter a, this indicates that the read/write head is placed over the cell containing a. The TM changes from one configuration to another according to its rules. For example, if the current configuration is $\#ws_i aw'\#$ and the TM has the rule $s_i a \longrightarrow s_j b$, this means that the read/write head positioned over the letter a will write b over it and change its state from s_i to s_j. The next configuration in the derivation will be thus $\#ws_j bw'\#$.

The TM halts with a word w iff there exists a derivation that, when started with the read/write head positioned at the beginning of w eventually reaches the final state (i.e. if $\#s_0 w\#$ derives $\#s_f\#$ by successive applications of the rewriting rules i–vii). The language $L(TM)$ accepted by TM consists of all words over the terminal alphabet T for which the TM halts. Note that TM is deterministic: at each step of the rewriting process, the application of at most one production is possible.

A FORMAL MODEL OF GENE REARRANGEMENT

In this section we describe several bio-operations we have studied as models of the homologous recombinations apparently underlying the process of gene unscrambling in ciliates [6–9]. We study these operations from a computational point of view. We namely summarize results in Kari and Kari [6] showing that, if the recognition of identical pointers is assumed to be sufficient to trigger recombination, the computational potential achieved is only that of finite automata.

We define circular words over Σ by declaring two words to be equivalent iff one is a cyclic permutation of the other. In other words, w is equivalent to w' iff they can be decomposed as $w = uv$ and $w' = vu$, respectively. Such a circular word $\bullet w$ refers to any of the circular permutations of the letters in w. Denote by $\Sigma^\bullet$ the set of all circular words over Σ.

DEFINITION I If $x \in \Sigma^+$ is a pointer, then the recombinations guided by x are defined as follows:

$$uxv + u'xv' \Longrightarrow uxv' + u'xv \text{ (linear/linear)} \tag{1}$$

$$uxvxw \Longrightarrow uxw + \bullet vx \text{ (linear/circular)} \tag{2}$$

$$\bullet\, uxv + \bullet u'xv' \Longrightarrow \bullet uxv'u'xv \text{ (circular/circular).} \tag{3}$$

(See figures in Kari and Kari [6].) Note that all recombinations in definition 1 are reversible; the operations can be performed also in the opposite directions.

For example, operation (2) models the process of intramolecular recombination (Figure 10.2). After the pointer x finds its second occurrence in $uxvxw$, the molecule undergoes a strand exchange in x that leads to the formation of two new molecules: uxw and a circular DNA molecule $\bullet vx$. Intramolecular recombination accomplishes the deletion of either sequence vx or xv from the original molecule $uxwxv$ and the positioning of w immediately next to ux.

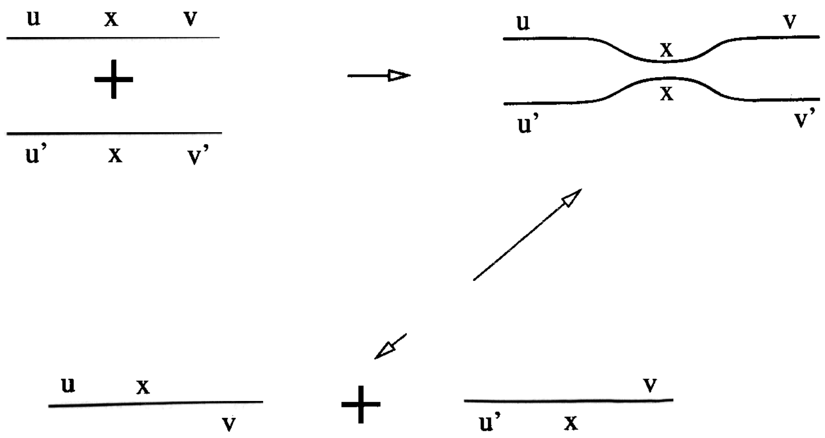

Figure 10.2 Linear/linear recombination.

This implies that operation 2 can be used to rearrange sequences in a DNA molecule, thus accomplishing gene unscrambling.

The above operations are similar to the the "splicing operation" introduced by Head [3] and circular splicing and mixed splicing [4, 14–16, 21]. It was subsequently shown that some of these models have the computational power of a universal Turing machine [1, 13, 22]. (See Head et al. [5] for a review.)

The process of gene unscrambling entails a series of successive or possibly simultaneous intra- and intermolecular homologous recombinations. This is followed by excision of all sequences $\tau_s y \tau_e$, where the sequence y is marked by the presence of telomere addition sequences τ_s for telomere "start" (at its 5' end), and τ_e for telomere "end" (at its 3' end). Thus, from a long sequence $u\tau_s y \tau_e v$, this step retains only $\tau_s y \tau_e$ in the macronucleus. Last, the enzyme telomerase extends the length of the telomeric sequences (usually double-stranded *TTTTGGGG* repeats in these organisms) from τ_s and τ_e to protect the ends of the DNA molecule.

We now make the assumption that, either by a structural alignment of the DNA or by other biochemical factors, the cell decides which sequences are non–protein-coding (IESs) and which are ultimately protein coding (MDSs), as well as which are the pointers x. Such biological shortcuts are presumably essential to bring into proximity the pointers x. Each of the n MDSs, denoted primarily by α_i, $1 \le i \le n$, is flanked by the pointers $x_{i-1,i}$ and $x_{i,i+1}$. Each pointer points to the MDS that should precede or follow α_i in the final sequence. The only exceptions are α_1, which is preceded by τ_s, and α_n, which is followed by τ_e in the input string or micronuclear molecule. Note that, although present generally once in the final macronuclear copy, each $x_{i,i+1}$ occurs at least twice in the micronuclear copy: once after α_i and once before α_{i+1}.

We denote by ϵ_k an internal sequence that is eliminated; ϵ_k does not occur in the final sequence. Thus, since unscrambling leaves one copy of each $x_{i,i+1}$ between α_i and α_{i+1}, an IES is nondeterministically either $\epsilon_k x_{i,i+1}$ or $x_{i-1,i}\epsilon_k$, depending on which pointer $x_{i,i+1}$ is eliminated. Similarly, an MDS is technically either $\alpha_i x_{i+1}$ or $x_{i-1,i}\alpha_i$. For this model, either choice is equivalent.

The following example (from Landweber and Kari [8]) models unscrambling of a micronuclear gene that contains MDSs in the scrambled order 2-4-1-3 using only the operation of linear/circular recombination:

$$\{u\ x_{12}\ \alpha_2\ x \quad \epsilon_1\ x_{34}\ \alpha_4\ \tau_e\ \epsilon_2\ \tau_s\ \alpha_1\ x \quad \epsilon_3\ x \quad \alpha_3\ x_{34}\ v\} \Longrightarrow$$

$$\{u\ x_{12}\ \epsilon_3\ x_{23}\ \alpha_3\ x_{34}\ v \qquad \bullet\alpha_2\ x \qquad\qquad \alpha_4\ \tau_e\ \epsilon_2\ \tau_s\ \alpha_1\ x_1$$

$$= \{u\ x_{12}\ \epsilon_3\ x_{23}\ \alpha_3\ x_{34}\ v \qquad \bullet\epsilon_1\ x_{34}\ \alpha_4\ \tau_e\ \epsilon_2\ \tau_s\ \alpha_1\ x_{12}\ \alpha_2\ x_{23}\} \Longrightarrow$$

$$\{u\ x_{12}\ \epsilon_3\ x_{23}\ \epsilon_1\ x_{34}\ \alpha_4\ \tau_e\ \epsilon_2\ \tau_s\ \alpha_1\ x_{12}\ \alpha_2\ x_{23}\ \alpha_3\ x_{34} \qquad \Longrightarrow$$

$$\{u\ x_{12}\ \epsilon_3\ x \qquad \bullet\alpha_4\ \tau_e\ \epsilon_2\ \tau_s\ \alpha_1\ x_{12}\ \alpha_2\ x \quad \alpha_3\ x_{34}\}$$

$$\{u\ x_{12}\ \epsilon_3\ x_{23}\ \epsilon_1\ x_{34}\ v \qquad \bullet\tau_s\ \alpha_1\ x_{12}\ \alpha_2\ x_{23}\ \alpha_3\ x_{34}\ \alpha_4 \qquad \Longrightarrow$$

$$\tau_s\ \alpha_1\ x_{12}\ \alpha_2\ x_{23}\ \alpha_3\ x \quad \alpha_4\ \tau \qquad \epsilon_2\ , \quad u\ x_{12}\ \epsilon_3\ x_2$$

The case in which all types of recombinations (linear/linear, linear/circular (Fig. 10.3), circular/circular (Fig. 10.4)) can occur, without restrictions, has been studied [6]. This study complements results obtained on linear splicing, circular splicing, self-splicing, and mixed splicing [5, 14–16, 21]. However, while theorem 2 may follow from a result in Head et al. [5] on the closure of an Abstract Family of Languages (AFL) under all splicings, theorem 1 characterizes the language, $L(R)$, of an arbitrary, context-free recombination system with a possibly infinite set of pointers and arbitrary axiom sets.

The intuitive image of context-free recombinations is that one can view strings as cables or extension cords with different types of plugs. Given a set of pointers J, each $x \in J$ defines one type of plug. Strings, both linear and circular, can then be viewed as consisting of elementary cables that only have plugs at their extremities. (A circular strand consists of elementary cables connected to form a loop.) A recombination step amounts to the following operations: take two connections using identical plugs (the connections can be in two different cables or in the same cable); unplug them; cross-plug to form new cables. We will assume, without loss of generality, that all sets of plugs J are subword-free [6].

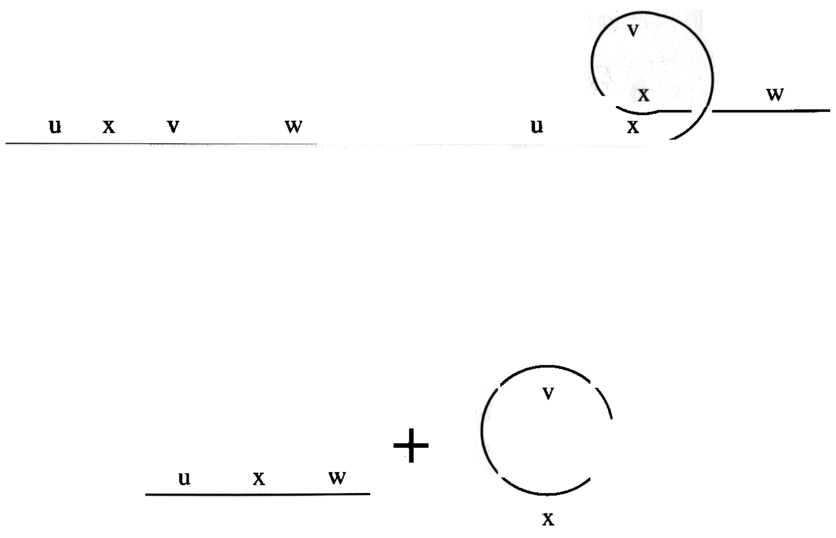

Figure 10.3 .inear/circular recombination.

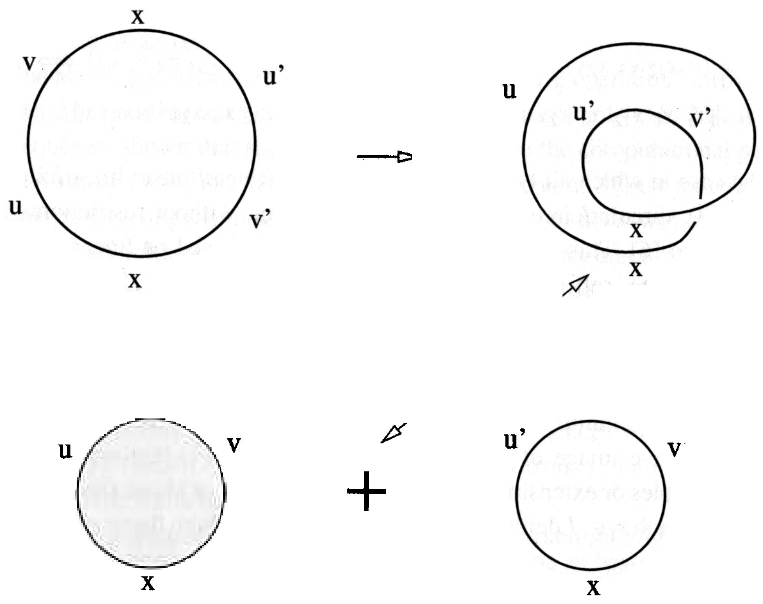

Figure 10.4 Circular/circular recombination.

DEFINITION 2 Let $J \subseteq \Sigma^+$ be a set of plugs. We define the set of elementary cables (respectively, left elementary cables and right elementary cables) with plugs in J as

$$E_J = (J\Sigma^+ \cap \Sigma^+ J) \setminus \Sigma^+ J \Sigma^+,$$

$$L_J = \Sigma^* J \setminus \Sigma^* J \Sigma^+,$$

$$R_J = J\Sigma^* \setminus \Sigma^+ J \Sigma^*$$

Note that an elementary cable in E_J is of the form $z_1 u = v z_2$, where $z_1, z_2 \in J$ are plugs. In other words, an elementary cable starts with a plug, ends with a plug, and contains no other plugs as subwords. The start and end plug can overlap.

A left elementary cable is of the form wz, where $z \in J$ is a plug and wz does not contain any other plug as a subword. In other words, if we scan wz from left to right, z is the first plug we encounter. Analogously, a right elementary cable is of the form zw, where $z \in J$ is a plug and wz does not contain any other plug as a subword.

DEFINITION 3 For a set of plugs $J \subseteq \Sigma^+$ and a linear word $w \in \Sigma^+$, the set of elementary cables with plugs in J occurring in w is defined as

$$E_J(w) = E_J \cap Sub(w)$$

while the set of left and right elementary cables occurring in w is

$$L_J(w) = L_J \cap Pref(w)$$

$$R_J(w) = R_J \cap Suff(w),$$

respectively, where $Pref(w)$, $Suff(w)$, $Sub(w)$ denote the set of all prefixes, suffixes and respectively subwords of w.

One can prove that recombination of cables does not produce additional elementary cables [6]. In other words, the set of the elementary cables of the result strings equals the set of elementary cables of the strings entering recombination.

A *context-free recombination system* is a construct whereby we are given a starting set of sequences and a list of pointers (plugs). New strings may be formed by recombinations among the existing strands: if given pointers are present, recombinations are performed as defined. Recombinations are context-free (i.e., they are not dependent on the context in which the pointers appear). The language of the system is defined as the set of all strands that can be thus obtained by repeated recombinations starting from the initial set.

DEFINITION 4 A context-free recombination system is a triple

$$R = (\Sigma, J, A)$$

where Σ is an alphabet and $J \subseteq \Sigma^+$ is a set of plugs, while $A \subseteq \Sigma^+ \cup \Sigma^\bullet$ is the set of axioms of the system.

The theorem below shows that a context-free recombination system characterized by a set of plugs J and a set of axioms A has the following property: any cable that consists of elementary cables plugged together after each other and that is either linear or circular can be obtained from the axioms using cross-plugging. Conversely, no other types of cables can be obtained from the axioms.

THEOREM I Let $R = (\Sigma, J, A)$ be a context-free recombination system. Then $L(R) = X$ where $X = \{w \in \Sigma^* \cup \Sigma^\bullet |$ either $E_J(w) = L_J(w) = R_J(w) = \emptyset$ and $w \in A$ or $E_J(w)$, $L_J(w)$, $R_J(w)$ are not all empty and $E_J(w) \subseteq E_J(A)$, $L_J(w) \subseteq L_J(A)$, $L_J(w) \subseteq L_J(A)\}$.

The theorem above will lead to the conclusion of this section provided we show that the language X is regular being accepted by a finite automaton. As X contains both linear and circular words, we have to first define the notion of acceptance of circular words by a finite automaton.

DEFINITION 5 Given a finite automaton $\mathcal{A}$, the circular language accepted by $\mathcal{A}$, denoted by $L(\mathcal{A})^\bullet$, is defined as the set of all words $\bullet w$ such that $\mathcal{A}$ has a cycle labeled by w.

The linear/circular language accepted by a finite automaton $\mathcal{A}$ is defined as $L(\mathcal{A}) \cup L(\mathcal{A})^\bullet$, where $L(\mathcal{A})$ is the linear language accepted by the automaton $\mathcal{A}$ defined in the usual way.

DEFINITION 6 A linear/circular language $L \subseteq \Sigma^* \cup \Sigma^\bullet$ is called *regular* if there exists a finite automaton $\mathcal{A}$ that accepts the linear and circular parts of L (i.e., that accepts $L \cap \Sigma^*$ and $L \cap \Sigma^\bullet$).

We can now formulate the main result presented in this section.

THEOREM 2 Let $J \subseteq \Sigma^*$ be a set of plugs and let $A \subseteq \Sigma^* \cup \Sigma^\bullet$ be a finite axiom set. Then the set X defined as in theorem 1 equals the linear/circular language accepted by a finite automaton $\mathcal{A}$ and is therefore regular.

Theorem 2 shows that the rewriting systems based on context-free recombinations are computationally weak, having only the power of finite automata. This is one more indicator that, most probably, the presence of pointers alone does not provide all the information needed for accurate splicing during gene rearrangement.

GUIDED RECOMBINATION SYSTEMS

As seen previously, the estimated running time of a pointer-search-and-match algorithm simulating gene rearrangement is prohibitively high. The previous section showed that the computational power of a formal computational model based on such context-free recombinations is very low—namely, that of finite automata. These and other biological arguments point to the fact that this model should be further refined to accurately reflect the biological reality [17]. We have introduced the additional assumption that homologous recombination is influenced by the presence of certain *guiding contexts* flanking the pointers present at the MDS-IES junctions [9]. The observed dependence on the old macronuclear sequence for correct IES removal in the distantly related ciliate *Paramecium* suggests that this is the case [11]. This restriction captures the fact that the pointers do not contain all the information for accurate splicing during gene unscrambling. In particular, we defined the notion of a *guided recombination system* based on operation 2 and proved that such systems have the computational power of a TM, the most widely used theoretical model of electronic computers [9].

We defined the contexts that restrict the use of recombinations by a *splicing scheme* [3, 4, 9] a pair $(\Sigma, \sim)$ where Σ 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 $(\Sigma, \sim)$, pairs $(p, x, q) \sim (p', x, q')$ now define the contexts necessary for a recombination between pointers x. Then we define *contextual intramolecular recombination* as

$$\{uxwxv\} \Longrightarrow \{uxv, \bullet wx\},$$

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').

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

$$\{uxv, \bullet wx\} \Longrightarrow \{uxwxv\},$$

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 $\sim$.

The operations defined in the preceding section are particular cases of contextual recombinations, where all the contexts are empty [i.e, $(\lambda, x, \lambda) \sim (\lambda, x, \lambda)$ for all $x \in \Sigma^+$]. This would correspond to the case where recombination may occur between every two pointers, regardless of their contexts.

DEFINITION 7 A guided recombination system is a construct $R = (\Sigma, \sim, A)$ where $(\Sigma, \sim)$ is a splicing scheme, and $A \in \Sigma^+$ is a linear string called the *axiom.*

Those strands which, by repeated contextual recombinations with initial and intermediate strands eventually produce the axiom, form the language of the guided recombination system, $L_a^k(R)$. $L_a^k(R)$ thus denotes the multiset of words $w \in \Sigma^*$ with the property that, if present initially in at least k copies, are able to produce the axiom A by a series of contextual recombinations. (A multiset is a set where to each element is associated a multiplicity. Operations applied to elements of a multiset change their multiplicities. After an operation, the multiplicities of the operation inputs decrease by one, while the multiplicity of the operation output increases by one. The need of multisets for modeling is justified in Landweber and Kari [9].)

THEOREM 3 Let L be a language over T^* accepted by TM $= (S, \Sigma \cup \{\#\}, P)$. Then there exist an alphabet Σ', a sequence $\pi \in \Sigma'^*$, depending on L, and a guided 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 idea of the proof is as follows. Consider that the rules of P are ordered in an arbitrary fashion and numbered. Thus, if TM has m rules, a rule is of the form $i : u_i \longrightarrow v_i$ where $1 \leq i \leq m$.

We construct a guided recombination system $R = (\Sigma', \sim, A)$ and a sequence $\pi \in \Sigma'^*$ with the required properties. The alphabet is $\Sigma' = S \cup \Sigma \cup \{\#\} \cup \{\$_i \mid 0 \leq i \leq m+1\}$. The axiom, i.e., the target string to be achieved at the end of the computation, consists of the final state of the TM bounded by markers:

$$A = \#^{n+2} s_f \ \#^{n+2} \$_0 \$_1 \ldots \$_m \$_{m+1},$$

where n is the maximum length of the left-side or right-side words of any of the rules of the TM.

The sequence π consists of the catenation of the right-hand sides of the TM rules bounded by markers, as follows:

$$\pi = \$_0 \ \$_1 e_1 v_1 f_1 \$_1 \ \$_2 e_2 v_2 f_2 \$_2 \ldots \$_m e_m v_m f_m \$_m \ \$_{m+1},$$

where $i : u_i \longrightarrow v_i$, $1 \leq i \leq m+1$ are the rules of TM and $e_i, v_i \in \Sigma \cup \{\#\}$.

If a word $w \in T^*$ is accepted by the TM, a computation starts then from a strand of the form $\#^{n+2} s_0 w \#^{n+2} \pi$, where we will refer to the subsequence starting with $\$_0$ as the "program" and to the subsequence at the left of $\$_0$ as the "data."

We construct the relation $\sim$ defining the contexts guiding the computations so that (1) the right-hand sides of rules of TM can be excised from the program as circular strands which then interact with the data; and (2) When the left-hand side of a TM rule appears in the data, the application of the rule can be simulated by the insertion of the circular strand encoding the right-hand side, followed by the deletion of the left-hand side.

With the help of these contexts, we can prove theorem 3—namely, that we can simulate the computation of any given TM by a series of contextual inter- and intramolecular recombinations.

Theorem 3 also 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, to simulate a computation of the TM 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 of operation 3 between circular strands or within a circular strand.

The proof that a guided recombination system can simulate a TM and thus any computation suggests that a functional macronuclear gene can be viewed as the output of a computation performed on the micronuclear sequence.

CONCLUSIONS

We have described a model for the process of gene rearrangement in spirotrich ciliates. Although the model is consistent with our limited knowledge of this biological process, it awaits rigorous testing by the tools of molecular genetics. The model in its present form is capable of universal computation. This hints at future directions and the use of ciliates as model systems for exploring cellular computation.

Acknowledgments This research was partially supported by the Natural Sciences and Engineering Research Council of Canada grant R2824AO1 to L.K. and National Institute of General Medical Sciences grant GM59708 to L.F.L. We thank Mark Daley for comments and Jeremy Newton-Smith with help in preparing the manuscript.

References

[1] E. Csuhaj-Varju, R. Freund, L. Kari, and G. Paun. DNA computing based on splicing: universality results. In L. Hunter and T. Klein, editors, *Proceedings of 1st Pacific Symposium on Biocomputing*, pages 179–190. World Scientific Publishing, Singapore, 1996.

[2] M. Daley. Complexity of gene unscrambling, 2002. Unpublished manuscript.

[3] T. Head. Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors. *Bull. Math. Biol.*, 49:737–759, 1987.

[4] T. Head. Splicing schemes and DNA. In G. Rozenberg and A. Salomaa, editors, *Lindenmayer Systems*, pages 371–383. Springer-Verlag, Berlin, 1991.

[5] T. Head and G. Paun, and D. Pixton. Language theory and molecular genetics. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, vol. 2, pages 295–358. Springer-Verlag, Berlin, 1997.

[6] J. Kari and L. Kari. Context-free recombinations. In C. Martin-Vide and V. Mitrana, editors, *Where Mathematics, Computer Science, Linguistics and Biology Meet*, pages 361–375. Kluwer, Dordrecht, The Netherlands, 2001.

[7] L. Kari, J. Kari, and L. Landweber. Reversible molecular computation in ciliates. In J. Karhumaki, H. Maurer, G. Paun, and G. Rozenberg, editors, *Jewels are Forever*, pages 353–363. Springer-Verlag, Berlin, 1999.

[8] L. F. Landweber and L. Kari. The evolution of cellular computing: nature's solution to a computational problem. *BioSystems*, 52(1–3):3–13, 1999.

[9] L. F. Landweber and L. Kari. Universal molecular computation in ciliates. In L. Landweber and E. Winfree, editors, *Evolution as Computation*. Springer-Verlag, Berlin, 2002.

[10] L. F. Landweber, T. Kuo, and E. Curtis. Evolution and assembly of an extremely scrambled gene. *Proc. Natl. Acad. Sci.*, 97(7):3298–3303, 2000.

[11] E. Meyer and S. Duharcourt. Epigenetic programming of developmental genome rearrangements in ciliates. *Cell*, 87:9–12, 1996.

[12] J. L. Mitcham, A. J. Lynn, and D. M. Prescott. Analysis of a scrambled gene: the gene encoding α-*telomere-binding* protein in *Oxytricha nova*. *Genes Devel.*, 6:788–800, 1992.

[13] G. Paun. On the power of the splicing operation. *Int. J. Comp. Math*, 59:27–35, 1995.