DIMACS Series in Discrete Mathematics
and Theoretical Computer Science

Computational Power of Gene Rearrangement

Lila Kari and Laura F. Landweber

ABSTRACT. In [8] we proposed a model to describe the homologous recombinations that take place during massive gene rearrangements in hypotrichous ciliates. Here we develop the model by introducing the dependency of homologous recombinations on the presence of certain contexts. We then prove that such a model has the computational power of a Turing machine. This indicates that, in principle, some unicellular organisms may have the capacity to perform any computation carried out by an electronic computer.

1. Introduction and notation

The process we model is gene rearrangement in ciliates, unicellular eukaryotes (nucleated cells) that 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 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. (See [**8**] for a review.)

The developing ciliate macronuclear "computer" (Figure 1) 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 in principle as splints, and the process of recombination results in linking the protein-encoding segments (MDSs) that belong next to each other in the final protein coding sequence. As such, the unscrambling of these protein-coding genes accomplishes an impressive feat of cellular computation. Other structural components of the ciliate chromatin presumably play a significant role, but the exact details of the mechanism remain elusive [**8**].

Before introducing the formal model, we summarize our notation. An alphabet Σ is a finite, nonempty set. A sequence of letters from Σ is called a string (word) over Σ and in our interpretation corresponds to a linear strand. The words are denoted by lowercase letters such as u, v, α_i, x_{ij}. The length of a word w is denoted by $|w|$ and represents the total number of occurrences of letters in the word. A

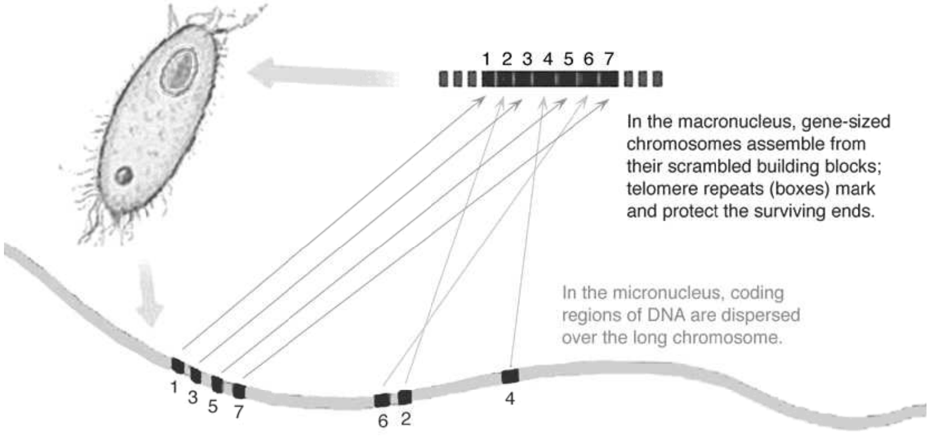

FIGURE 1. **Overview of gene unscrambling.** Dispersed coding MDSs 1-7 reassemble during macronuclear development to form the functional gene copy (top), complete with telomere addition to mark and protect both ends of the gene. (From [**8**].)

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^+. We also define circular words over Σ by declaring two words to be equivalent if and only if (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.

A rewriting system $TM = (S, \Sigma \cup \{\#\}, P)$ is called a *Turing machine*, [**12**], iff:

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

(ii) 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.

(iii) The productions (rewriting rules) of P are of the forms

(1) $s_i a \longrightarrow s_j b$ (overprint)
(2) $s_i ac \longrightarrow a s_j c$ (move right)
(3) $s_i a\# \longrightarrow a s_j B\#$ (move right and extend workspace)
(4) $c s_i a \longrightarrow s_j ca$ (move left)
(5) $\# s_i a \longrightarrow \# s_j Ba$ (move left and extend workspace)
(6) $s_f\ a \longrightarrow s_f$
(7) $a\ s_f \longrightarrow s_f$

where s_i and s_j are states 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 (2) and (3) (resp.(4) and (5)) or else contains both (3) and (2) for every c (resp.contains both (5) and (4) 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 (1), (3), (5).

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 Turing machine 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 succesive applications of the rewriting rules (1) - (7). 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.

2. Computational power of gene rerrangement

In this section we define the notion of a *guided recombination system* 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 [**8**] and illustrated in Figure 2 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 ([**9**]). This restriction captures the fact that the guide sequences do not contain all the information for accurate splicing during gene unscrambling.

Using an approach developed in [**7**] we use contexts to restrict the use of recombinations. A *splicing scheme*, [**3**], [**4**] is 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 the repeats x. Then we define *contextual intramolecular recombination* as

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

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'.

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

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

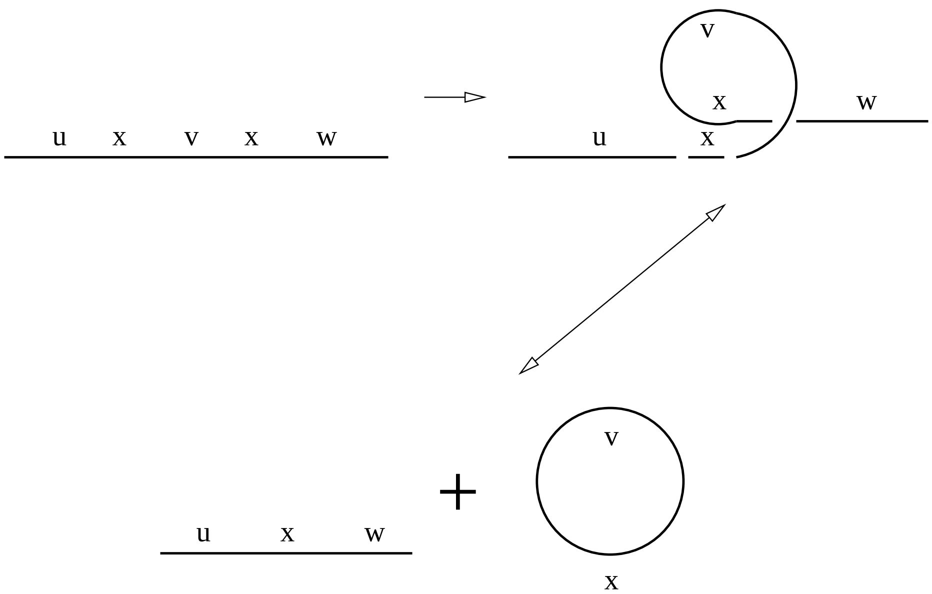

FIGURE 2. Intra- and intermolecular recombinations using repeats x. During intramolecular recombination, after x finds its second occurrence in $uxvxw$, the molecule undergoes a strand exchange in x that leads to the formation of two new molecules: a linear DNA molecule uxw and a circular one $\bullet vx$. The reverse operation is intermolecular recombination.

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 above operations resemble the "splicing operation" introduced by Head in [**3**] and "circular splicing" ([**4**], [**13**], [**11**]). [**10**], [**1**] and subsequently [**14**] showed that these models have the computational power of a universal Turing machine. (See [**5**] for a review.)

The operations defined in [**8**] are particular cases of guided recombination, where all the contexts are empty, i.e, $(\lambda, x, \lambda) \sim (\lambda, x, \lambda)$ for all $x \in \Sigma^+$. 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 have proved that they can only generate regular languages.

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 [**2**], if $\mathbf{N}$ is the set of natural numbers, a multiset of Σ^* is a mapping $M : \Sigma^* \longrightarrow \mathbf{N} \cup \{\infty\}$, where, for a word $w \in \Sigma^*$, $M(w)$ represents the number of occurrences of w. Here, $M(w) = \infty$ means that there are unboundedly many copies of the string w. The set $\mathrm{supp}(M) = \{w \in \Sigma^* | \; M(w) \neq 0\}$, the *support of* M, consists of the strings that are present at least once in the multiset M.

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

A guided recombination system R defines a *derivation relation* 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 $\Sigma^* \cup \Sigma^\bullet$, we say that S derives S' and we write $S \Longrightarrow_R S'$, iff one of the following two cases hold:

(1) there exist $\alpha \in \mathrm{supp}(S)$, $\beta, \bullet\gamma \in \mathrm{supp}(S')$ such that

– $\{\alpha\} \Longrightarrow \{\beta, \bullet\gamma\}$ 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 $\alpha', \bullet\beta' \in \mathrm{supp}(S)$, $\gamma' \in \mathrm{supp}(S')$ such that

– $\{\alpha', \bullet\beta'\} \Longrightarrow \{\gamma'\}$ 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 of the guided recombination system. Formally,

$$L_a^k(R) = \{w \in \Sigma^* | \;\; \{w\} \Longrightarrow_R^* S \text{ and } A \in \mathrm{supp}(S)\},$$

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. *Let* L *be a language over* T^* *accepted by a Turing machine* $TM = (S, \Sigma \cup \{\#\}, P)$ *as above. Then there exist an alphabet* Σ', *a sequence* $\pi \in \Sigma'^*$, *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* $k \geq 1$.

Proof. 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 $\Sigma' = S \cup \Sigma \cup \{\#\} \cup \{\$_i | \; 0 \leq i \leq m+1\}$. 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 Turing machine.

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 $i:\ u_i \longrightarrow v_i$, $1 \le i \le m+1$ 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 $\sim$ so that

(i) The right-hand sides of rules of TM can be excised from the program as circular strands which then interact with the data.

(ii) 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.

To accomplish *(i)*, for each rule $i: u \longrightarrow v$ of the TM, we introduce in $\sim$ the pairs

$$(C) \qquad (\$_{i-1}, \$_i, evf) \sim (evf, \$_i, \$_{i+1}),$$

for all $e, f \in \Sigma \cup \{\#\}$.

To accomplish *(ii)* for each rule $i: u \longrightarrow v$ of the TM, add to the relation $\sim$ the pairs

$$(A) \qquad (ceu, f, d) \sim (\$_i ev, f, \$_i ev),$$

$$(B) \qquad (c, e, uf\$_i) \sim (uf\$_i, e, vfd),$$

for all $c \in \{\#\}^*\Sigma^*$, $d \in \Sigma^*\{\#\}^*$, $|c| = |d| = n$, $e, f \in \Sigma \cup \{\#\}$.

Following the above construction of the alphabet Σ', sequence π and recombination system R, for any $x, y \in \Sigma'$ we can simulate a derivation step of the TM as follows:

$$\{xceufdy\$_0 \ldots \$_{i-1}\$_i evf\$_i\$_{i+1} \ldots \$_{m+1}\} \Longrightarrow_R$$

$$\{xceufdy\$_0 \ldots \$_{i-1}\$_i\$_{i+1} \ldots \$_{m+1}, \bullet\$_i evf\} \Longrightarrow_R$$

$$\{xceuf\$_i evfdy\$_0 \ldots \$_{i-1}\$_i\$_{i+1} \ldots \$_{m+1}\} \Longrightarrow_R$$

$$\{xcevfdy\$_0 \ldots \$_{i-1}\$_i\$_{i+1} \ldots \$_{m+1},\ \bullet\$_i euf\}.$$

The first step is an intramolecular recombination using contexts (C) around the repeat $\$_i$ to excise $\bullet\$_i evf$. Note that if the current strand does not contain a subword $\$_i evf\$_i$, this can be obtained from another copy of the original linear strand, which is initially present in k copies. The second step is an intermolecular recombination using contexts (A) around the repeat f, to insert $\$_i evf$ after $ceuf$. The third step is an intramolecular recombination using contexts (B) around the direct repeat e to delete $\$_i euf$ from the linear strand. Thus, the "legal" insertion/deletion succession that simulates one TM derivation step claims that any u in the data, that is surrounded by at least $n+1$ letters on both sides may be replaced by v. This explains why in our choice of axiom we needed $n+1$ extra symbols $\#$ to provide the contexts allowing recombinations to simulate all TM rules, including (3) and (5).

From the fact that a TM derivation step can be simulated by recombination steps we deduce that, if the TM accepts a word w, then we can start a derivation in R from

$$\#^{n+2}s_0w\#^{n+2}\pi = \#^{n+2}s_0w\#^{n+2}\$_0\$_1\ldots\$_ie_iv_if_i\$_i\ldots\$_m\$_{m+1}$$

and reach the axiom by only using recombinations according to R. This means that our word is accepted by R, that is, it belongs to $L_a^k(R)$ for some k. Note that if some rules of the TM have not been used in the derivation then they can be excised in the end, and that k should be large enough so that we do not exhaust the set of rewriting rules.

For the converse implication, it suffices to prove that starting from the strand $\#^{n+2}s_0w\#^{n+2}\pi$, no other recombinations except those that excise rules of TM from the program and those that simulate steps of the TM in the data are possible in R.

In the beginning of the derivation we start with no circular strands and k copies of the linear strand

$$\#^{n+2}s_0w\#^{n+2}\$_0\ldots\$_ie_iv_if_i\$_i\ldots\$_{m+1},\, w\in T^*,$$

where $i : u_i \longrightarrow v_i$ are TM rules, $e_i, f_i \in \Sigma\cup\{\#\}$, $1\le i\le m$.

Assume now that the current multiset contains linear strands of the form $\delta_0\pi$, where $\delta_0\in\Sigma'^*$ contains only one state symbol and no $\$_i$ symbols and

$$\pi = \$_0r_1r_2\ldots r_m\$_{m+1},$$

with r_i either encoding the right-hand side of a rule or being the remnant of a rule, i.e., $r_i\in\{\$_ie_iv_if_i\$_i\}\cup\{\$_i\}$, $1\le i\le m$. Moreover, assume that the circular strands present in the multiset are of the form $\bullet\$_ie_iv_if_i$, with e_i, v_i, f_i as before.

Then:

(i) We cannot use (A) or (B) to insert or delete in the program because that would require the presence of strands $ceufd$ or $\$_iev f\$_iev$ (if we want to use (A)) or $ceuf\$_i$ or $uf\$_ievfd$ (if we want to use (B)). However none of these strands can appear in the program. Indeed, the 1st, 3rd, and 4th word all contain subwords over $\Sigma\cup\{\#\}$ of length at least $n+3$, and this is more than the length of the longest subword over $\Sigma\cup\{\#\}$ present in the program. The 2nd word cannot appear in the program because no marker $\$_i$ appears alone in p, as p contains always at least two consecutive markers.

(ii) We cannot use (C) to insert or delete in the data because that would require the presence in δ_0 of two consecutive markers $\$_{i-1}\$_i$ or $\$_i\$_{i+1}$, which contradicts our assumptions.

(iii) We cannot use (C) to insert in the program because that would require the presence of a circular strand with two markers, - contradiction with our assumptions.

Arguments *(i)* - *(iii)* show that the only possible recombinations are either deletions in the program using (C), which result in the release of circular strands $\bullet\$_ievf$, or insertions/deletions in the data using (A) and (B).

Assuming that the data contains as a subword the left-hand side of a TM rule $i : u \longrightarrow v$, and assuming that the necessary circular strand $\bullet\$_ievf$ has already been excised from the program, the next step is to show that the only possible insertions/deletions in the data are those simulating a rewriting step of TM using rule i.

Indeed, in this situation,

(1) It is not possible to delete in δ_0 using (A), or insert or delete using (B), as all these operations would require a $\$_i$ in δ_0. Therefore only an insertion in δ_0 using (A) is possible. An insertion according to (A) may only take place between a sequence $ceuf$ and a sequence d, where u contains a state symbol, i.e. the read/write head, c and d have length n and e and f are letters. This means that, for the insertion to take place, the linear word has to be of the form

$$\delta_0\ \pi = xceufdy\ \pi$$

and the intermolecular recombination with the circular strand $\bullet\$_i evf$ inserts $\$_i evf$ between u and f producing the linear strand

$$\delta_1\ \pi = xceuf\$_i evfdy\ \pi.$$

Note that, as δ_0 contains only one state symbol and no marker $\$_i$, the newly formed word δ_1 contains only two state symbols (read/write heads), one in u and one in v, and only one marker $\$_i$. (Here we use the fact that every rule $u \longrightarrow v$ of the TM has exactly one state symbol on each side.)

(2) Starting now from $\delta_1\pi$,

(2a) We can delete in δ_1 using (B) and, as there is only one $\$_i$ in δ_1, there is only one position where the deletion can happen. After the release of the strand $\bullet\$_i euf$ as a circular strand, the linear strand produced is

$$\delta_2\ \pi = xcevfdy\ \pi.$$

(2b) No insertion in δ_1 using (A) may take place, as the marker $\$_i$ "breaks" the contexts necessary for further insertions.

Indeed, the occurrence of another insertion according to (A) requires that the read/write head symbol be both followed and preceded by at least $(n+1)$ letters different from $\$_i$. In δ_1, the first read/write head is in u and the number of letters following it is at most $|u|-1+|f| \leq n-1+1=n$, which is not enough as a right context for insertion using (A). The second read/write head is in v and the number of letters preceding it is at most $|e|+|v|-1 \leq 1+n-1=n$, which is not enough as a left context for insertion using (A).

(2c) No deletion in δ_1 using (A) may occur, as this would require the presence of a repeat f bordered by a $\$_i ev$ on each side. This would imply that the current strand δ_1 contains two markers $\$_i$, which is not true.

(2d) No insertion in δ_1 using (B) is possible, as that would require the presence of a circular strand containing $\$_i evfd$. The length of such a strand would be at least $1+|e|+|v|+|f|+|d|$ that is, at least $n+4$, which is more than the length of any initial or intermediate circular strand. Indeed, all the circular strands produced from the program have length $n+3$ and the only circular strands that are released are, as seen in (2a), of the form $\bullet\$_i euf$ and thus also have lengths at most $n+3$.

The arguments above imply that the only possible operations on the data simulate legal rewritings of the TM by tandem recombination steps that necessarily follow each other.

Together with the arguments that the only operations affecting the program are excisions of circular strands encoding TM rules, and that the circular TM rules do not interact with each other, this proves the converse implication.

From the definition of the Turing machine we see that n, the maximum length of a word occurring in a TM rule, equals 4, which completes the proof of the theorem.

Q.E.D.

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 [**8**].

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.

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.

In conclusion, we have developed a model for the acrobatic 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.

Acknowledgements. Jarkko Kari for essential contribution to the proof of the theorem in its present form. Rani Siromoney, Gilles Brassard for suggestions, Erik Winfree and Gheorghe Păun for comments. Grzegorz Rozenberg, Richard Lipton, David Prescott and Hans Lipps for discussion, Mark Daley for Figure 2.

References

[1] Csuhaj-Varju, E., Freund, R., Kari, L., and G. Paun. 1996. 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. Pages 179-190.

[2] Eilenberg, S., 1984. *Automata, Languages and Machines.* Academic Press, New York.

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

[4] Head, T. (1991). Splicing schemes and DNA. In *Lindenmayer systems* (Rozenberg, G. and Salomaa, A., Eds.). Springer Verlag, Berlin. Pages 371-383.

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

[6] Hoffman, D.C., and D.M. Prescott. 1997. Evolution of internal eliminated segments and scrambling in the micronuclear gene encoding DNA polymerase α in two *Oxytricha* species. *Nucl. Acids Res.* 25: 1883-1889.

[7] Kari, L., and G. Thierrin. 1996. Contextual insertion/deletions and computability. *Information and Computation* 131: 47-61.

[8] Landweber, L.F., Kari, L. 1998. The evolution of cellular computing: nature's solution to a computational problem. Proceedings of 4th DIMACS meeting on DNA based computers, Philadephia. Pages 3-15.

[9] Meyer, E. and Duharcourt, S. 1996. Epigenetic Programming of Developmental Genome Rearrangements in Ciliates. *Cell* 87 : 9-12.

[10] Păun, G. 1995. On the power of the splicing operation. *Int. J. Comp. Math* 59 : 27-35.

[11] Pixton, D., 1995. Linear and circular splicing systems. Proceedings of the First International Symposium on Intelligence in Neural and Biological Systems, IEEE Computer Society Press, Los Alamos. Pages 181-188.

[12] Salomaa, A. 1973. *Formal Languages.* Academic Press, New York.

[13] Siromoney, R., Subramanian, K.G. and Rajkumar Dare, V. 1992. Circular DNA and splicing systems. In Parallel Image Analysis. Lecture Notes in Computer Science 654, Springer Verlag, Berlin. Pages 260-273.

[14] Yokomori, T., Kobayashi, S., and Ferretti, C. 1997. Circular Splicing Systems and DNA Computability. In Proc. of IEEE International Conference on Evolutionary Computation'97. Pages 219-224.

(L. Kari) Department of Computer Science, University of Western Ontario, London, ON, N6A 5B7 Canada

E-mail address: `lila@csd.uwo.ca`

URL: `http://www.csd.uwo.ca/~ lila`

(L. Landweber) Department of Ecology and Evolutionary Biology, Princeton University, NJ 08544-1003 USA

E-mail address: `lfl@princeton.edu`

URL: `http://www.princeton.edu/~ lfl`