Modelling simple operations for gene assembly

Tero Harju\textsuperscript{1,3}, Ion Petre\textsuperscript{2,3}, and Grzegorz Rozenberg\textsuperscript{4}

\textsuperscript{1} Department of Mathematics, University of Turku
Turku 20014 Finland
harju@utu.fi

\textsuperscript{2} Department of Computer Science, Åbo Akademi University
Turku 20520 Finland
ipetre@abo.fi

\textsuperscript{3} Turku Centre for Computer Science
Turku 20520 Finland

\textsuperscript{4} Leiden Institute for Advanced Computer Science, Leiden University
Niels Bohrweg 1, 2333 CA Leiden, the Netherlands, and
Department of Computer Science, University of Colorado at Boulder
Boulder, Co 80309-0347, USA
rozenber@liacs.nl

Summary. The intramoelcular model (Ehrenfeucht et al., 2001) for gene assembly
in ciliates considers three operations, \( h, \) \( h_i, \) and \( h_d \) that can assemble any micronu-
clear gene pattern through folding and recombination: the molecule is folded so that
two occurrences of a pointer (short nucleotide sequence) get aligned and then the
sequence is rearranged through recombination of pointers. In general, the sequence
rearranged by one operation can be arbitrarily long and may consist of many coding
and non-coding blocks. We consider in this paper some restricted variants of the
three operations, where only one coding block is rearranged at a time. We present
in this paper the molecular model of these simple operations. We also introduce a
mathematical model for the simple operations, on three levels of abstractions: on
MDS descriptors, signed permutations, and signed double occurrence strings. Interest-
ingly, we show that simple assemblies possess rather involved properties: a gene
pattern may have both successful and unsuccessful assemblies and also more than
one successful strategy.

1.1 Introduction

The \textit{stichotrichous} ciliates have a very unusual way of organizing their genomic
sequences. In the macronucleus, the somatic nucleus of the cell, each gene is a
contiguous DNA sequence. Genes are generally placed on their own very short
DNA molecules. In the micronucleus, the germline nucleus of the cell, the genes
are placed on long chromosomes separated by noncoding material. However, the
same gene is completely differently organized than in the macronucleus: it

\[ \text{DNA molecule} \]
is broken into pieces called MDSs (macronuclear destined sequences) that are separated by noncoding blocks called IESs (internally eliminated sequences). Moreover, the order of MDSs is shuffled and some MDSs may be inverted. The ciliates have several copies of the macronucleus (all identical to each other) and several micronuclei (all identical to each other) — the exact number of copies depends on the species. During sexual reproduction, ciliates destroy the old macronucleus and transform a micronucleus into a new macronucleus. In this process, ciliates must assemble all macronuclear genes by placing in the orthodox order all MDSs to yield a functional macronuclear gene. To this aim they are using pointers, short nucleotide sequences that identify each MDS. Thus, each MDS $M$ begins with a pointer that is exactly repeated in the end of the MDS preceding $M$ in the orthodox order. The ciliates use the pointers to splice together all MDSs in the correct order.

The intramolecular model for gene assembly, introduced in [10] and [28] consists of three operations: $l$, $h$, and $d$. In each of these operations, the macronuclear chromosome folds on itself so that two or more pointers get aligned and through recombination, two or more MDSs get combined into a bigger composite MDS. The process continues until all MDSs have been assembled. For details related to ciliates and gene assembly we refer to [16], [21], [22], [23], [24], [25], [26], [27]. For details related to the intramolecular model and its mathematical formalizations we refer to [4], [5], [8], [9], [13], [14], [15], [29], [30], as well as to the recent monograph [6]. For a different intermolecular model we refer to [18], [19], [20].

There are no restrictions in general on the number of nucleotides between the two pointers that should be aligned in a certain fold. However, all available experimental data are consistent with restricted versions of our operations, in which between two aligned pointers there is at most one MDS, see [6], [7], and [12]. We propose in this paper a mathematical model for the simple variants of $l$, $h$, and $d$. The model is in terms of MDS descriptors, signed permutations, and signed double occurrence strings.

1.2 Mathematical preliminaries

For an alphabet $\Sigma$ we denote by $\Sigma^*$ the set of all finite strings over $\Sigma$. For a string $u$ we denote $\text{dom}(u)$ the set of letters occurring in $u$. We denote by $\lambda$ the empty string. For strings $u, v$ over $\Sigma$, we say that $u$ is a substring of $v$, denoted $u \preceq v$, if $v = xuy$, for some strings $x, y$.

Let $\Sigma_n = \{1, 2, \ldots, n\}$ and let $\Sigma_n = \{\emptyset, 1, 2, \ldots, \pi\}$ be a signed copy of $\Sigma_n$. For any $i \in \Sigma_n$ we say that $i$ is an unsigned letter, while $\pi$ is a signed letter.

For a string $u$ over $\Sigma_n \cup \Sigma_n$, $u = a_1a_2\ldots a_m$, we denote its inversion by $\overline{u} = \overline{a_1a_2\ldots a_m}$, where $\overline{a} = a$, for all $a \in \Sigma_n$.

A (unsigned) permutation $\pi$ over an interval $\Delta = \{i, i+1, \ldots, i+l\}$ is a bijective mapping $\pi : \Delta \to \Delta$. We often identify $\pi$ with the string $\pi(i) \pi(i + 1) \ldots \pi(i + l)$. We say that $\pi$ is (cyclically) sorted if $\pi = k(k + 1)\ldots i + l$.
1 Modelling simple operations for gene assembly

An operation \( i (i + 1) \ldots (k - 1) \), for some \( i \leq k \leq i + l \). A signed permutation over \( \Delta \) is a string \( \psi \) over \( \Delta \cup \bar{\Delta} \) such that \( ||\psi|| \) is a permutation over \( \Delta \). We say that \( \psi \) is cyclically sorted if \( \psi = \psi' (k + 1) \ldots i + l i (i + 1) \ldots (k - 1) \) or \( \psi = (k - 1) \ldots (i + 1) i (i + 1) \ldots (k + 1) \), for some \( i \leq k \leq i + l \). Equivalently, \( \psi \) is sorted if either \( \psi \) or \( \psi' \) is a sorted unsigned permutation. In the former case we say that \( \psi \) is sorted in the orthodox order, while in the latter case we say that \( \psi \) is sorted in the inverted order.

There is rich literature on sorting (signed and unsigned) permutations, both in connection to their applications to computational biology in topics such as genomic rearrangements or genomic distances, but also as a classical topic in discrete mathematics, see, e.g., [1], [2], [11], [17].

1.3 The intramolecular model

We present in this section the intramolecular model: the folds and the recombinations for each of the operations \( \text{id} \), \( \text{hi} \), and \( \text{diad} \), as well as their simple variants.

1.3.1 The structure of micronuclear genes

A micronuclear gene is broken into coding blocks called MDSs (macronuclear destined sequences), separated by non-coding blocks called IESs (internally-eliminated sequences). In the macronucleus however, all MDSs are spliced together into contiguous coding sequences, with no IESs present anymore. It is during gene assembly that ciliates eliminate IES and splice MDSs together. A central role in this process is played by pointers, relatively short nucleotide sequences at both ends of each MDS. As it turns out, the pointer in the end of the \((i - 1)\)st MDS (in the order given by the macronuclear gene sequence), say \( M_{i-1} \), coincides as a nucleotide sequence with the pointer in the beginning of the \(i\)th MDS, say \( M_i \), for all \( i \).

Based on these observation, we can represent the micronuclear genes by their sequences of MDSs only. E.g.,, we represent the structure of the micronuclear gene encoding the actin protein in *Sterkiella nova* by the sequence of MDSs \( M_3 M_4 M_5 M_6 M_7 M_8 M_9 \), where we indicate that the second MDS, \( M_2 \), is inverted in the micronucleus. Moreover, in some cases, we represent each MDS by its pair of pointers: we denote by \( i \) the pointer in the beginning of the \(i\)th MDS \( M_i \). Thus, MDS \( M_i \) can be represented by its pair of pointers as \((i, i + 1)\). The first and the last MDSs are special: \( M_1 = (b, 2) \) and \( M_k = (k, e) \), where \( b \) and \( e \) are special beginning/ending markers. In this case, the gene in Figure 1.1 is represented as \((3, 4)(4, 5)(6, 7)(5, 6)(7, 8)(9, e)[3, 7](b, 2)(8, 9)\). One more simplification can also be made. The gene may be represented by the sequence of its pointers only, thus ignoring the markers and the parenthesis above – this representation still gives enough information to trace the gene assembly process. Details on model forming can be found in [6].
1.3.2 Three molecular operations

Three molecular operations, ld, hi, dlad were conjectured in [10] and [28] for gene assembly. In each of them, the micronuclear genome folds on itself in such a way that certain types of folds may be formed and recombination may take place, see Figure 1.2. It is important to note that all foldings are aligned by pointers, some relatively short nucleotide sequences at the intersection of MDSs and IESs. The pointer at the end of an MDS $M$ coincides (as a nucleotide sequence) with the pointer in the beginning of the MDS following $M$ in the assembled gene. We refer for more details to [6].

![Diagram of molecular operations](image)

**Fig. 1.2.** Illustration of the ld, hi, dlad molecular operation showing in each case: (i) the folding, (ii) the recombination, and (iii) the result.

It is known that ld, hi, and dlad can assemble any gene pattern or, in other words, any sequence of MDSs can be transformed into an assembled MDS $(b,c)$ (in which case we say that it has been assembled in the *orthodox* order) or $(E,F)$ (we say it has been assembled in the *inverted* order), see [6] and [7] for formal proofs.

1.3.3 Simple operations for gene assembly

Note that all three operations ld, hi, and dlad are *intramolecular*, that is, only one molecule folds on itself to rearrange its coding blocks. For a different, intermolecular model for gene assembly, see, [18], [19], and [20].

Since ld excises one circular molecule, that molecule can only contain non-coding blocks (or, in a special case, contain the entire gene, see [6] for details.
on boundary ld; we say that ld must always be *simple* in a successful assembly. As such, the effect of ld is that it will combine two consecutive MDSs into a bigger composite MDS. E.g., consider that $M_iM_{i+1}$ is part of the molecule, i.e., MDS $M_{i+1}$ succeeds $M_i$ being separated by one IES $I$. Thus, pointer $i + 1$ has two occurrences that flank $I$: one in the end of MDS $M_i$ and the other one in the beginning of MDS $M_{i+1}$. Then ld makes a fold as in Figure 1.2:kl(i) aligned by pointer $i + 1$, excises IES $I$ as a circular molecule and combines $M_i$ and $M_{i+1}$ into a longer coding block as shown in Figure 1.2:kl(ii)-kl(iii).

In the case of hi and dlad, the rearranged sequences may be arbitrarily large. E.g., in the actin I gene in S. nova, see Figure 1.1, pointer 3 has two occurrences: one in the beginning of $M_3$ and one, inverted, in the end of $M_2$. Thus, hi is applicable to this sequence with the hairpin aligned on pointer 3, even though five MDSs separate the two occurrences of pointer 3. Similarly, dlad is applicable to the MDS sequence $M_2M_8M_6M_5M_1M_7M_3M_{10}M_9M_4$, with the double loops aligned on pointers 3 and 5. Here the first two occurrences of pointers 3, 5 are separated by two MDSs ($M_8$ and $M_6$) and their second occurrences are separated by four MDSs ($M_3$, $M_{10}$, $M_9$, $M_4$).

It turns out however that all available experimental data, see [3], are consistent with applications of the so-called “simple” hi and dlad: particular instances of hi and dlad where the folds and thus, the rearranged sequences contain only one MDS. We define the simple operations in the following.

**Fig. 1.3.** The MDS/IES structures where the *simple hi*-rule is applicable. Between the two MDSs there is only one IES.

**Fig. 1.4.** The MDS/IES structures where *simple dlad*-rules is applicable. Straight line denotes one IES.

An application of the hi-operation on pointer $p$ is *simple* if the part of the molecule that separates the two copies of $p$ in an inverted repeat contains only one MDS (and one IES). We have here two cases, depending on whether the first occurrence of $p$ is incoming or outgoing. The two possibilities are
illustrated in Figure 1.3, where the MDs are indicated by rectangles and their flanking pointers are shown.

An application of ddad on pointers $p, q$ is simple if the sequence between the first occurrences of $p, q$ and the sequence between the second occurrences of $p, q$ consist of either one MD or one IES. We have again two cases, depending on whether the first occurrence of $p$ is incoming or outgoing. The two possibilities are illustrated in Figure 1.4.

Recall that an operation ld is always simple (by definition) in the intramolecular model so that no coding sequence is lost.

One immediate property of simple operations is that they are not universal, i.e., there are sequences of MDs that cannot be assembled by simple operations. One such example is the sequence $(\mathcal{T}, \mathcal{B})(4, e)(3, 4)(2, 3)$. Indeed, neither ld, nor simple hi, nor simple ddad is applicable to this sequence.

### 1.4 Formal models for simple operations

We introduce in this section a formal model for simple operations. The model is expressed on three level of abstraction: on MDs descriptors, signed permutations, and signed double occurrence strings.

#### 1.4.1 Modelling by MDs descriptors

As noted above, micromolecular gene patterns may be represented by the sequence of their MDs, while MDs may be represented only by the pair of their flanking pointers, ignoring the rest of the sequences altogether. Indeed, since the pointers align all the folds required by gene assembly and guide the splicing of MDs, the whole process can be tracked even with this (remarkable) simplification. Thus, an MD $M_i$ is represented as $(i, i + 1)$, while its inversion is denoted as $(i, i + 1)$. A sequence of such pairs will be called MD descriptor and will be used to represent the structure of micromolecular genes.

We define the notion formally in the following.

Let $\mathcal{M} = \{b, e, \overline{b}, \overline{e}\}$ be the set of markers and their inversions, and $\Pi_\kappa = \{2, 3, \ldots, \kappa\} \cup \{\mathcal{T}, \mathcal{B}\}$ the set of pointers and their inversions, where $\kappa$ is the number of MDs in the gene of interest. In the following, $\kappa$ is an arbitrary but fixed nonnegative integer.

Let then

$$\Gamma_\kappa = \{(b, e), (\overline{b}, \overline{e}), (b, i), (\overline{b}, \overline{i}), (i, e), (\overline{i}, \overline{e}) \mid 2 \leq i \leq \kappa\}$$

$$\cup \{(i, j) \mid 2 \leq i < j \leq \kappa\}.$$ 

For each $x \in \Pi_\kappa \cup \mathcal{M}$, let

$$\tilde{x} = \begin{cases} 1, & \text{if } x \in \{b, \overline{b}\}, \\ \kappa + 1, & \text{if } x \in \{e, \overline{e}\}, \\ ||x||, & \text{if } x \in \Pi_\kappa. \end{cases}$$
For each \( \delta = (x, y) \in \Gamma_\kappa \), let \( \delta = \min\{\tilde{x}, \tilde{y}\} \), \( \max\{\tilde{x}, \tilde{y}\} - 1 \).

**Example 1.** Let \( \delta = (4, 5)(5, 8)(b, 4)(8, e)(5, 6) \). Then the pairs occurring in \( \delta \) have the following values: \( (4, 5) = [4, 4], \ (5, 8) = [6, 7], \ (b, 4) = [1, 3], \ (8, e) = [8, 8] \) and \( (5, 6) = [5, 5] \).

Consider \( \delta \in \Gamma^*_\kappa \), \( \delta = \delta_1\delta_2 \ldots \delta_n \), with \( \delta_i \in \Gamma_\kappa \) for each \( i \). We say that \( \delta \) is an MDS descriptor if the intervals \( \delta_i \), for \( i = 1, 2, \ldots, n \), form a partition of the interval \([1, \kappa + 1]\).

For each micromolecule gene pattern, its associated MDS descriptor is obtained by denoting each MDS by its pair of pointers or markers.

**Example 2.** The MDS descriptor associated to gene actin in S.nova, see Figure 1.1 is \((3, 4) (4, 5) (6, 7) (5, 6) (7, 8) (9, e) (3, 2) (b, 2) (8, 9) \).

We can now define the simple operations as rewriting rules on MDS descriptors in accordance with the molecular model shown in Figures 1.3 and 1.4.

1. For each pointer \( p \in \Pi_\kappa \), the ld-rule for \( p \) is defined as follows:

\[
\text{ld}_p(\delta_1(q, p)(p, r)\delta_2) = \delta_1(q, r)\delta_2, \quad (\ell 1)
\]

where \( q, r \in \Pi_\kappa \cup M \), \( m_1, m_2 \in M \) and \( \delta_1, \delta_2 \in \Gamma^*_\kappa \).

2. For each pointer \( p \in \Pi_\kappa \), the sh-rule for \( p \) is defined as follows:

\[
\text{sh}_p(\delta_1(p, q)\Psi\delta_2) = \delta_1(\Psi, \delta_2), \quad (h 1)
\]

where \( q, r \in \Pi_\kappa \cup M \), and \( \delta_i \in \Gamma^*_\kappa \), for each \( i = 1, 2, 3 \).

3. For each pointers \( p, q \in \Pi_\kappa \), the sd-rule for \( p, q \) is defined as follows:

\[
\text{sd}_{p, q}(\delta_1(p, q)\delta_2(p, q)\delta_3) = \delta_1(p, q)\delta_2(p, q)\delta_3, \quad (d 1)
\]

where \( r_1, r_2 \in \Pi_\kappa \cup M \), and \( \delta_i \in \Gamma^*_\kappa \), for each \( i = 1, 2, 3 \).

For an MDS descriptor \( \delta \) and operations \( \varphi_1, \ldots, \varphi_n, n \geq 1 \), a composition \( \varphi = \varphi_n \ldots \varphi_1 \) is an assembly strategy for \( \delta \), if \( \varphi \) is applicable to \( \delta \). Also, \( \varphi \) is successful for \( \delta \) if either \( \varphi(\delta) = (b, e) \) (in which case we say that \( \delta \) has been assembled in the orthodox order) or \( \varphi(\delta) = (\Psi, b) \) (and we say that \( \delta \) has been assembled in the inverted order).

**Example 3.** The actin gene in S.nova may be assembled by simple operations as follows. If \( \delta = (3, 4) (4, 5) (6, 7) (5, 6) (7, 8) (9, e) (3, 2) (b, 2) (8, 9) \), then
$l_{\delta}(\delta) = (3, 5) \ (6, 7) \ (5, 6) \ (7, 8) \ (9, e) \ (3, 7) \ (b, 2) \ (8, 9)$

$s_{d, \delta}(l_{\delta}(\delta)) = (3, 7) \ (7, 8) \ (9, e) \ (3, 7) \ (b, 2) \ (8, 9)$

$l_{\gamma}(s_{d, \delta}(l_{\delta}(\delta))) = (3, 8) \ (9, e) \ (3, 7) \ (b, 2) \ (8, 9)$

$s_{d, \delta}(l_{\gamma}(s_{d, \delta}(l_{\delta}(\delta)))) = (3, e) \ (3, 7) \ (b, 2)$

$s_{\gamma}(s_{d, \delta}(l_{\gamma}(s_{d, \delta}(l_{\delta}(\delta))))) = (e, \gamma) \ (b, 2)$

$s_{\gamma}(s_{\gamma}(s_{d, \delta}(l_{\gamma}(s_{d, \delta}(l_{\delta}(\delta))))) = (e, \gamma) \ (b, 2)$

1.4.2 Modelling by signed permutations

The gene structure of a ciliate can also be represented as a signed permutation, denoting the sequence and orientation of each MDS, while omitting all IEs. E.g., the signed permutation associated to gene actin 1 in S. novia is $3 \ 4 \ 6 \ 5 \ 7 \ 9 \ 2 \ 1 \ 8$. The rearrangements made by $l_{\delta}$, $h_{\delta}$, $d_{\delta}$ at the molecular level leading to bigger composite MDSs have a correspondent on permutations in combining two already sorted blocks into a longer sorted block. Assembling a gene is equivalent in terms of permutations to sorting the permutation associated to the micronuclear gene as detailed below. Indeed, the gene is assembled once all MDSs are placed in the correct order.

When formalizing the gene assembly as a sorting of permutations we will effectively ignore the operation $l_{\delta}$ observing that once such an operation becomes applicable to a gene pattern, it can be applied at any later step of the assembly, see [4] and [8] for a formal proof. In particular, we can assume that all $l_{\delta}$ operations are applied in the last stage of the assembly, once all MDSs are sorted in the correct order. In this way, the process of gene assembly can indeed be described as a process of sorting the associated signed permutation, i.e., arranging the MDSs in the proper order, be that orthodox or inverted.

It is worth noting that the signed permutations are equivalent with the MDS descriptors as far as their expressibility is concerned. Indeed, the mapping $\psi$ defined so that $\psi(i) = (i, i + 1)$, for all $1 < i < \kappa$, $\psi(1) = (b, 2)$, $\psi(e) = (e, c)$ is a bijective morphism between the set of signed permutations and that of MDS descriptors. Some differences do exist when modelling gene assembly with descriptors or permutations. E.g., modelling the assembly with MDS descriptors is a rewriting process of eliminating pointers, leading ultimately to an assembled descriptors with no pointers. On this level, we can keep track of every pointer in the gene assembly – this is often useful. The down side is that the descriptors introduce a tedious mathematical notation and reasoning about them is typically involved. The signed permutations on the other hand represent an elegant, classical topic in mathematics and a large literature about them exists. Gene assembly on permutations becomes a process of sorting signed permutations, a well-studied topic in the literature. An additional technical advantage here is that the base alphabet of the permutation does not change through the process as it is the case with the
descriptors. The down side of the signed permutations is that they do not
denote the pointers explicitly.

The molecular model of simple operations in Figures 1.3 and 1.4 can be
formalized as a sorting of signed permutations as follows.

(2') For each \( p \geq 1 \), \( sh_p \) is defined as follows:

\[
sh_p(x(p + 1) \ldots (p + k + 1) y) = x(p + k + 1) \ldots (p + 1) y,
sh_p(x(p - k) \ldots (p + 1) y) = x(p - k) \ldots p(p + 1) y,
sh_p(x(p + 1) \ldots p y) = x(p + 1) \ldots (p - k) y,
\]

where \( k \geq 0 \) and \( x, y, z \) are signed strings over \( \Sigma_n \). We denote \( Sh = \{ sh_i | 1 \leq i \leq n \} \).

(3') For each \( p, 2 \leq p \leq n - 1 \), \( sd_p \) is defined as follows:

\[
sd_p(x(p - i) \ldots p y(p - i - 1) (p + 1) z) = x y(p - i - 1) (p - i) \ldots p(p + 1) z,
sd_p(x(p - i - 1) (p + 1) y(p - i) \ldots p z) = x(p - i - 1) (p - i) \ldots p(p + 1) y z,
\]

where \( i \geq 0 \) and \( x, y, z \) are signed strings over \( \Sigma_n \). We also define \( sd_p^\pi \) as
follows:

\[
sd_p(x(p + 1)(p - i - 1) y(p - i) \ldots (p - i) z) = x(p + 1) y(p - i) \ldots (p - i) z,
sd_p(x(p - i) (p + 1)(p - i - 1) z) = x y(p + 1) z,
\]

where \( i \geq 0 \) and \( x, y, z \) are signed strings over \( \Sigma_n \). We denote \( Sd = \{ sd_i, sd_i^\pi | 1 \leq i \leq n \} \).

We say that a signed permutation \( \pi \) over the set of integers \( \{ i, i+1, \ldots, i+l \} \) is
sortable if there are operations \( \phi_1, \ldots, \phi_k \in Sh \cup Sd \) such that \( \phi_1 \circ \ldots \circ \phi_k \mid \pi \) is a (cyclically) sorted permutation. We also say in this case
that \( \phi_1 \circ \ldots \circ \phi_k \) is a sorting strategy for \( \pi \). We say that \( \pi \) is \( Sh \)-sortable
if \( \phi_1, \ldots, \phi_k \in Sh \) and we say that \( \pi \) is \( Sd \)-sortable if \( \phi_1, \ldots, \phi_k \in Sd \). A
composition \( \phi \) is called a unsuccessful strategy for \( \pi \) if \( \phi(\pi) \) is an unsorable
permutation.

Example 4. (i) Permutation \( \pi_1 = 347612 \) is sortable and a sorting strategy
is \( Sh(\phi(\pi_1)) = 345612 \). Permutation \( \pi'_1 = 345612 \) is unsorable.
Indeed, no \( sh \) operations and no \( sd \) operation is applicable to \( \pi'_1 \).

(ii) Permutation \( \pi_2 = 13427 \) is sortable and has only one sorting strategy:
\( sh(\phi(\pi_2)) = 12345 \).

(iii) There exist permutations with several successful strategies, even leading
to different sorted permutations. One such permutation is \( \pi_3 = 35124 \).
Indeed, \( sd(\pi_3) = 51234 \). At the same time, \( sd(\pi_3) = 34512 \).
(iv) The simple operations yield a nondeterministic process: there are permutations having both successful and unsuccessful sorting strategies. One such permutation is \( \pi_4 = 135792468 \). Note that \( sd_3(sd_5(sd_7(\pi_4))) = 1923456789 \) is a sortable permutation. However, \( \pi_4 \) can be sorted, e.g., by the following strategy: \( sd_4(sd_6(sd_8(\pi_4))) = 123456789 \).

(v) Permutation \( \pi_5 = 13524 \) has both successful and unsuccessful sorting strategies. Indeed, \( sd_3(\pi_5) = 15234 \), a sortable permutation. However, \( sd_2(sd_4(\pi_5)) = 12345 \) is sorted.

(vi) Applying a cyclic shift to a permutation may render it sortable. Indeed, permutation \( 21435 \) is sortable, while \( 52143 \) is not.

(vii) Consider the signed permutation \( \pi_7 = 11139572413615810121416 \). Operation \( sd \) may be applied to \( \pi_7 \) on integers \( 3, 6, 9, 11, 13, \) and \( 15 \). Doing that however leads to a sortable permutation:

\[
sd_3(sd_0(sd_0(sd_11(sd_13(sd_15(\pi_7)))))) = 15672348910111213141516
\]

However, omitting \( sd_3 \) from the above composition leads to a sorting strategy for \( \pi_7 \): let

\[
\pi_7' = sd_0(sd_0(sd_11(sd_13(sd_15(\pi_7)))))) = 13567248910111213141516
\]

Then \( sd_2(sd_4(\pi_7')) \) is a sorted permutation.

(viii) Consider the signed permutation \( \pi_8 \) associated to the actin gene in S. nova, \( \pi_8 = 346579218 \). A sorting strategy for \( \pi_8 \) is shown below, compare also with Example 3:

\[
\begin{align*}
sd_5(\pi_8) &= 345679218 \\
sd_8(sd_5(\pi_8)) &= 345678921 \\
sd_2(sd_8(\pi_8)) &= 587654321 \\
sd_1(sd_2(\pi_8)) &= 587654321
\end{align*}
\]

1.4.3 Modelling by signed double occurrence strings

The structure of a gene may be simplified by representing only the sequence of its pointer, see [4], [6], and [8]. Indeed, since the assembled gene has no pointers anymore and all the operations are based on the sequence and orientation of pointers, such a simplification is possible. The strings we obtain are called signed double occurrence strings and are defined in the following.

Let \( \Sigma \) be an alphabet and \( \Sigma^* \) its signed copy. A string \( v \in (\Sigma \cup \Sigma^\ast) \) is a signed double occurrence string if for every letter \( a \in \text{dom}(v) \), \( v \) has exactly two occurrences from the set \( \{a, \overline{a}\} \). We also say then that \( v \) is a legal string. If \( v \) contains both substrings \( a \) and \( \overline{a} \), then \( a \) is positive in \( u \); otherwise, \( a \) is negative in \( u \).

Example 5. Consider the signed string \( u = 24375345 \) over \( \Delta_5 \). Clearly, \( u \) is legal. Pointers 2 and 5 are positive in \( u \), while 3 and 4 are negative in \( u \). On
the other hand, the string \( w = 243735 \) is not legal, since 4 has only one occurrence in \( w \).

We can associate a unique legal string to any gene pattern by writing its sequence of pointers only. Formally, we can define the following mapping: 
\[
\mu((i, j)) = i, \quad \text{for all } 2 \leq i < j \leq \kappa, \quad \mu((k, k)) = i, \quad \text{for all } 2 \leq k \leq \kappa, \quad \text{and } \mu((k, e)) = \lambda. \quad \text{Then } \mu \text{ defines a morphism from the set of MDS descriptors to the set of legal string. We say that } \mu(\delta) \text{ is the legal string associated to } \delta.
\]

**Example 6.** The MDS descriptor associated to the actin gene in *S. nova* is \( \delta = (3, 4)(4, 5)(6, 7)(5, 6)(7, 8)(9, e) (\text{ } 3, 7)(b, 2)(8, 9) \). Its legal string, obtained from \( \delta \) by writing only the sequence of pointers and their orientations is 3445675678937289.

We refer to [6] for the formalization of the intramolecular model on the level of legal strings. We only define in the following the simple operations as rewriting rules on legal strings. Without risk of confusion, we will use the notation ld, sh and sd also for legal strings.

The simple operations can be defined as rewriting rules on legal strings as follows.

1. For each pointer \( p \in \Pi_{\kappa} \), the ld-rule for \( p \) is defined as follows:
   \[
   \text{ld}_p(u_1 p u_2) = u_1 u_2.
   \]
   where \( u_1, u_2 \in (\Sigma \cup \overline{\Sigma})^* \). Let \( \text{ld} = \{ \text{ld}_p \mid p \in \Pi_{\kappa}, \ k \geq 2 \} \).

2. For each pointer \( p \in \Pi_{\kappa} \), the sh-rule for \( p \) is defined as follows:
   \[
   \text{sh}_p(u_1 p u_2 \overline{u}_3) = u_1 \overline{u}_2 u_3,
   \]
   where \( u_1, u_2, u_3 \in (\Sigma \cup \overline{\Sigma})^* \) and \( |u_2| \leq 1 \). Let \( \text{sh} = \{ \text{sh}_p \mid p \in \Pi_{\kappa}, \ k \geq 2 \} \).

3. For each pointers \( p, q \in \Pi_{\kappa} \), the sd-rule for \( p, q \) is defined as follows:
   \[
   \text{sd}_{pq}(u_1 p q u_2 p q u_3) = u_1 u_2 u_3,
   \]
   where \( u_1, u_2, u_3 \in (\Sigma \cup \overline{\Sigma})^* \). Let \( \text{sd} = \{ \text{sd}_{pq} \mid p, q \in \Pi_{\kappa}, \ k \geq 2 \} \).

A composition \( \varphi = \varphi_n \ldots \varphi_1 \) of operations from \( \text{ld} \cup \text{sh} \cup \text{sd} \) is a string reduction of \( u \), if \( \varphi \) is applicable to \( u \). Also, \( \varphi \) is successful for \( u \) if \( \varphi(u) = \lambda \), the empty string.

**Example 7.** The signed double occurrence string associated to the actin gene in *S. nova* is \( u = 3445675678937289 \). A successful reduction of \( u \) using only simple operations is the following, compare with Examples 3 and 4(viii):
\( \text{ld}_4(u) = 3567567893289 \)
\( \text{sd}_{5,6}(\text{ld}_4(u)) = 3789232 \)
\( \text{ld}_{5,6}(\text{ld}_4(u)) = 3893289 \)
\( \text{sd}_{8,9}(\text{ld}_7(\text{sd}_{5,6}(\text{ld}_4(u)))) = 3522 \)
\( \text{sh}_2(\text{sd}_{8,9}(\text{ld}_7(\text{sd}_{5,6}(\text{ld}_4(u))))) = \lambda. \)

1.5 Problems

We introduced in this paper the molecular model of the so-called simple operations, a restricted variant of the intramolecular model for gene assembly. In these simple operations, the type of fold that a micronuclear chromosome has to make in the assembly is very restricted: only one MDS is moved during the subsequent recombination. While this variant is not universal anymore, it is still powerful enough to assemble all known micronuclear gene patterns up to date. As such, it is natural to presume, at least for now, that ciliates may be using only this restricted model. Several relevant questions can then be asked. One of the most important ones is: what are the gene patterns that can be assembled using the simple operations? Also, we noticed that, while the simple model is not universal anymore, it remains non-deterministic: there are gene patterns that have both successful and unsuccessful assembly strategies. Deciding if a given pattern may be assembled by simple operations and finding/characterizing its successful strategies is another important problem. From a computational point of view, a study of the complexity of the simple assemblies seems very interesting. A study of the simple operations was already initiated in [12].

References


