TUCS

Tero Harju | Ion Petre | Grzegorz Rozenberg

# Modelling simple operations for gene assembly

Turku Centre *for* Computer Science

TUCS Technical Report
No 697, August 2005

# Modelling simple operations for gene assembly

Tero Harju
Department of Mathematics, University of Turku
Turku Centre for Computer Science
Turku 20014 Finland
`harju@utu.fi`

Ion Petre
Academy of Finland and
Department of Computer Science, Åbo Akademi University
Turku Centre for Computer Science
Turku 20520 Finland
`ipetre@abo.fi`

Grzegorz Rozenberg
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`

TUCS Technical Report
No 697, August 2005

## Abstract

The intramolecular model (Ehrenfeucht et al, 2001) for gene assembly in ciliates considers three operations, ld, hi, and dlad that can assemble any micronuclear 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: MDS descriptors, signed permutations, and signed double occurrence strings. Interestingly, 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.

**TUCS Laboratory**
Computational Biomodelling
Discrete Mathematics for Information Technology

## 1 Introduction

The *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 genes in the micronucleus are organized completely differently than in the macronucleus: a micronuclear gene 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 (compared to their order in the macronuclear version of a given gene) may be shuffled and some MDSs may be inverted. The ciliates may 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 macronuclei and transform a micronucleus into a new macronucleus. In this process, ciliates must assemble all micronuclear genes by placing in the proper (orthodox) order all MDSs to yield a functional macronuclear gene. *Pointers*, short nucleotide sequences that identify each MDS, play an important role in the process. 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: $\mathsf{ld}$, $\mathsf{hi}$, and $\mathsf{dlad}$. In each of these operations, the micronuclear 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]. In this paper we propose a mathematical model that takes this restriction into account by considering "simple" variants of $\mathsf{ld}$, $\mathsf{hi}$, and $\mathsf{dlad}$. The model is formulated in terms of MDS descriptors, signed permutations, and signed double occurrence strings.

## 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 by $\mathsf{dom}(u)$ 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 $u \leq v$, if $v = xuy$, for some strings $x, y$.

Let $\Sigma_n = \{1, 2, \ldots, n\}$ and let $\overline{\Sigma}_n = \{\overline{1}, \overline{2}, \ldots, \overline{n}\}$ be a *signed copy* of $\Sigma_n$. For any $i \in \Sigma_n$ we say that $i$ is a *unsigned letter*, while $\overline{i}$ is a *signed* letter. For a string $u = a_1 a_2 \ldots a_m$ over $\Sigma_n \cup \overline{\Sigma}_n$, its inversion $\overline{u}$ is defined by $\overline{u} = \overline{a}_m \ldots \overline{a}_2 \overline{a}_1$, where $\overline{\overline{a}} = a$, 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 $\pi = k\,(k+1)\ldots i+l\,i\,(i+1)\ldots(k-1)$, for some $i \leq k \leq i+l$. A *signed permutation* over $\Delta$ is a string $\psi$ over $\Delta \cup \overline{\Delta}$ such that $\|\psi\|$ is a permutation over $\Delta$. We say that $\psi$ is *(cyclically) sorted* if $\psi = k\,(k+1)\ldots i+l\,i\,(i+1)\ldots(k-1)$ or $\psi = \overline{(k-1)}\ldots\overline{(i+1)}\,\overline{i}\,\overline{(i+l)}\ldots\overline{(k+1)}\,\overline{k}$, for some $i \leq k \leq i+l$. Equivalently, $\psi$ is sorted if either $\psi$, or $\overline{\psi}$ 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].

## 3 The intramolecular model

We present in this section the intramolecular model: the folds and the recombinations for each of the operations ld, hi, and dlad, as well as their simple variants.

### 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 $M_3\,M_4\,M_6\,M_5\,M_7\,M_9\,\overline{M}_2\,M_1\,M_8$, 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, and so $M_1$ is represented by $(b, 2)$ and $M_k$ by $(k, e)$, where $b$ and $e$ are special beginning/ending markers. In this case, the gene in Figure 1 is represented as $(3,4)(4,5)(6,7)(5,6)(7,8)(9,e)(\overline{3},\overline{2})(b,2)(8,9)$. 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].

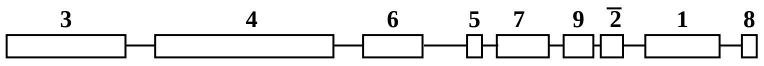

Figure 1: Structure of the micronuclear gene encoding actin protein in *Sterkiella nova.*

## 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 2. It is important to note that all foldings are aligned by pointers. We refer for more details to [6].

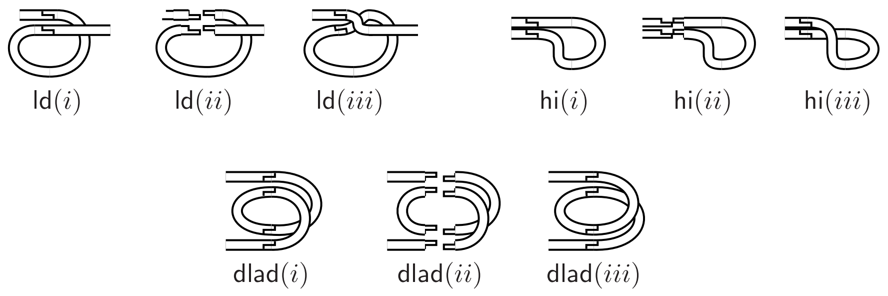

Figure 2: Illustration of the ld, hi, dlad molecular operation showing in each case: (i) the folding, (ii) the recombination, and (iii) the result.

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 $(b, e)$ (in which case we say that it has been assembled in the *orthodox* order) or $(\overline{e}, \overline{b})$ (we say it has been assembled in the *inverted* order), see [6] and [7] for formal proofs.

## 3.3 Simple operations for gene assembly

Note that all three operations ld, hi, dlad are *intramolecular*, that is, a 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 a 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 2:ld(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 2:ld(ii)-ld(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, 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.

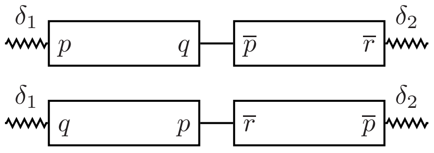

Figure 3: The MDS/IES structures where the *simple hi*-rule is applicable. Between the two MDSs there is only one IES.

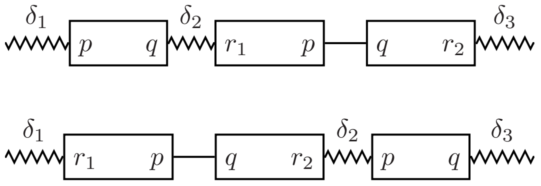

Figure 4: The MDS/IES structures where *simple dlad*-rules is applicable. Straight line denotes one IES.

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 3, where the MDSs are indicated by rectangles and their flanking pointers are shown.

An application of dlad 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 MDS 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 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 MDSs that cannot be assembled by simple operations. One such example is the sequence $(\overline{2}, \overline{b})(4, e)(3, 4)(2, 3)$. Indeed, neither ld, nor simple hi, nor simple dlad is applicable to this sequence.

## 4 Formal models for simple operations

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

## 4.1 Modelling by MDS descriptors

As noted above, micronuclear gene patterns may be represented by the sequence of their MDSs, while MDSs may be represented only by the pair of their flanking pointers, ignoring the rest of the sequences altogether. Indeed, since all the folds required by gene assembly are aligned on pointers, and the splicing of MDSs takes place through pointers, the whole process can be tracked even with this (remarkable) simplification. Thus, an MDS $M_i$ is represented as$(i, i+1)$, while its inversion is denoted as $(\overline{i+1}, \overline{i})$. A sequence of such pairs will be called MDS descriptor and will be used to represent the structure of micronuclear 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 $\Pi_\kappa = \{2, 3, \ldots, \kappa\} \cup \{\overline{2}, \overline{3}, \ldots, \overline{\kappa}\}$ 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

$$\begin{aligned}\Gamma_\kappa = &\{\, (b,e), (\overline{e},\overline{b}), (b,i), (\overline{i},\overline{b}), (i,e), (\overline{e},\overline{i}) \;\mid\; 2 \le i \le \kappa \,\}\\ &\cup\; \{\, (i,j) \;\mid\; 2 \le i < j \le \kappa \,\}.\end{aligned}$$

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

$$\widehat{x} = \begin{cases} 1, & \text{if } x \in \{b, \overline{b}\},\\ \kappa + 1, & \text{if } x \in \{e, \overline{e}\},\\ \|x\|, & \text{if } x \in \Pi_\kappa. \end{cases}$$

For each $\delta = (x, y) \in \Gamma_\kappa$, let $\widehat{\delta} = [\min\{\widehat{x}, \widehat{y}\}, \max\{\widehat{x}, \widehat{y}\} - 1]$.

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

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 $\widehat{\delta_i}$, for $i = 1, 2, \ldots, n$, form a partition of the interval $[1, \kappa + 1]$.

For each micronuclear 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, is $(3,4)\,(4,5)\,(6,7)\,(5,6)\,(7,8)\,(9,e)\,(\overline{3},\overline{2})\,(b,2)\,(8,9)$.

We can now define the simple operations as rewriting rules on MDS descriptors in accordance with the molecular model shown in Figures 3 and 4.

(1) For each pointer $p \in \Pi_\kappa$, the $\mathsf{ld}$-*rule* for $p$ is defined as follows:

$$\mathsf{ld}_p(\delta_1(q,p)(p,r)\delta_2) = \delta_1(q,r)\delta_2, \tag{$\ell$1}$$

$$\mathsf{ld}_p((p,m_1)(m_2,p)) = (m_2,m_1), \tag{$\ell$2}$$

where $q, r \in \Pi_\kappa \cup \mathcal{M}$, $m_1, m_2 \in \mathcal{M}$ 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:

$$\mathsf{sh}_p(\delta_1(p,q)(\overline{p},\overline{r})\delta_2) = \delta_1(\overline{q},\overline{r})\delta_2, \tag{h1}$$
$$\mathsf{sh}_p(\delta_1(q,p)(\overline{r},\overline{p})\delta_2) = \delta_1(q,r)\delta_2, \tag{h2}$$

where $q, r \in \Pi_\kappa \cup \mathcal{M}$, 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:

$$\mathsf{sd}_{p,q}(\delta_1(p,q)\delta_2(r_1,p)(q,r_2)\delta_2) = \delta_1\delta_2(r_1,r_2)\delta_3, \tag{d1}$$
$$\mathsf{sd}_{p,q}(\delta_1(r_1,p)(q,r_2)\delta_2(p,q)\delta_3) = \delta_1(r_1,p)(q,r_2)\delta_2(p,q)\delta_3, \tag{d2}$$

where $r_1, r_2 \in \Pi_\kappa \cup \mathcal{M}$, 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 $\varphi = \varphi_\kappa \ldots \varphi_1$ 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 $\varphi(\delta) = (\overline{e}, \overline{b})$ (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 $\delta = (3,4)\,(4,5)\,(6,7)\,(5,6)\,(7,8)\,(9,e)\,(\overline{3},\overline{2})\,(b,2)\,(8,9)$, then

$$\begin{aligned}
&\mathsf{ld}_4(\delta) = (3,5)\,(6,7)\,(5,6)\,(7,8)\,(9,e)\,(\overline{3},\overline{2})\,(b,2)\,(8,9)\\
&\mathsf{sd}_{5,6}(\mathsf{ld}_4(\delta)) = (3,7)\,(7,8)\,(9,e)\,(\overline{3},\overline{2})\,(b,2)\,(8,9)\\
&\mathsf{ld}_7(\mathsf{sd}_{5,6}(\mathsf{ld}_4(\delta))) = (3,8)\,(9,e)\,(\overline{3},\overline{2})\,(b,2)\,(8,9)\\
&\mathsf{sd}_{8,9}(\mathsf{ld}_7(\mathsf{sd}_{5,6}(\mathsf{ld}_4(\delta)))) = (3,e)\,(\overline{3},\overline{2})\,(b,2)\\
&\mathsf{sh}_3(\mathsf{sd}_{8,9}(\mathsf{ld}_7(\mathsf{sd}_{5,6}(\mathsf{ld}_4(\delta))))) = (\overline{e},\overline{2})\,(b,2)\\
&\mathsf{sh}_2(\mathsf{sh}_3(\mathsf{sd}_{8,9}(\mathsf{ld}_7(\mathsf{sd}_{5,6}(\mathsf{ld}_4(\delta)))))) = (\overline{e},\overline{b}).
\end{aligned}$$

## 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 IESs. E.g., the signed permutation associated to gene actin I in S.nova is $3\,4\,6\,5\,7\,9\,\overline{2}\,1\,8$. The rearrangements made by ld, hi, dlad at the molecular level leading to bigger composite MDSs correspond to permutations that combine two already sorted blocks into a longer sorted block. Thus, in the framework of permutations, assembling a gene is equivalent to sorting the permutation associated to the micronuclear gene as exaplined bellow. 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 ld 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 ld 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 $1 < i < \kappa$, $\psi(1) = (b, 2)$, and $\psi(\kappa) = (\kappa, e)$

is a bijective morphism between the set of signed permutations and the set 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 assembled descriptors with no pointers. On this level, we can keep track of every pointer in the gene assembly – this is often useful. The downside 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 topic that is well-studied 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 downside of the signed permutations is that they do not denote the pointers explicitly.

The molecular model of simple operations in Figures 3 and 4 can be formalized as a sorting of signed permutations as follows.

(2') For each $p \geq 1$, $\mathsf{sh}_p$ is defined as follows:

$$\begin{aligned}
&\mathsf{sh}_p(x\,(p+1)\ldots(p+k+1)\,\overline{p}\,y) = x\,(\overline{p+k+1})\ldots(\overline{p+1})\,\overline{p}\,y,\\
&\mathsf{sh}_p(x\,\overline{p}\ldots(\overline{p-k})(p+1)\,y) = x\,(p-k)\ldots p\,(p+1)\,y,\\
&\mathsf{sh}_p(x\,p\,(\overline{p+k+1})\ldots\overline{p+1}\,y) = x\,p\,(p+1)\ldots(p+k+1)\,y,\\
&\mathsf{sh}_p(x\,(\overline{p+1})(p-k)\ldots p\,y) = x\,(\overline{p+1})\,\overline{p}\ldots(\overline{p-k})\,y,
\end{aligned}$$

where $k \geq 0$ and $x, y, z$ are signed strings over $\Sigma_n$. We denote $\mathsf{Sh} = \{\mathsf{sh}_i \mid 1 \leq i \leq n\}$.

(3') For each $p$, $2 \leq p \leq n-1$, $\mathsf{sd}_p$ is defined as follows:

$$\begin{aligned}
&\mathsf{sd}_p(x\,(p-i)\ldots p\,y\,(p-i-1)\,(p+1)\,z) = x\,y\,(p-i-1)\,(p-i)\ldots p\,(p+1)\,z,\\
&\mathsf{sd}_p(x\,(p-i-1)\,(p+1)\,y\,(p-i)\ldots p\,z) = x\,(p-i-1)\,(p-i)\ldots p\,(p+1)\,y\,z,
\end{aligned}$$

where $i \geq 0$ and $x, y, z$ are signed strings over $\Sigma_n$. We also define $\mathsf{sd}_{\overline{p}}$ as follows:

$$\begin{aligned}
&\mathsf{sd}_{\overline{p}}(x\,(\overline{p+1})(\overline{p-i-1})\,y\,\overline{p}\ldots(\overline{p-i})\,z) = x\,(\overline{p+1})\,\overline{p}\ldots(\overline{p-i})\,(\overline{p-i-1})\,y\,z,\\
&\mathsf{sd}_{\overline{p}}(x\,\overline{p}\ldots(\overline{p-i})\,y\,(\overline{p+1})\,(\overline{p-i-1})\,z) = x\,y\,(\overline{p+1})\,\overline{p}\ldots(\overline{p-i})\,(\overline{p-i-1})\,z,
\end{aligned}$$

where $i \geq 0$ and $x, y, z$ are signed strings over $\Sigma_n$. We denote $\mathsf{Sd} = \{\mathsf{sd}_i, \mathsf{sd}_{\overline{i}} \mid 1 \leq i \leq n\}$.

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

**Example 4.** (i) Permutation $\pi_1 = 3\,\overline{4}\,\overline{5}\,6\,\overline{1}\,2$ is sortable and a sorting strategy is $\mathsf{sh}_1(\mathsf{sh}_4(\mathsf{sh}_3(\pi_1))) = 3\,4\,5\,6\,1\,2$. Permutation $\pi_1' = 3\,4\,5\,6\,\overline{1}\,\overline{2}$ is unsortable. Indeed, no $\mathsf{sh}$ operations and no $\mathsf{sd}$ operation is applicable to $\pi_1'$.

(ii) Permutation $\pi_2 = 1\,3\,4\,2\,\overline{5}$ is sortable and it has only one sorting strategy: $\mathsf{sh}_4(\mathsf{sd}_2(\pi_2)) = 1\,2\,3\,4\,5$.

(iii) There exist permutations with several successful strategies, even leading to different sorted permutations. One such permutation is $\pi_3 = 3\,5\,1\,2\,4$. Indeed, $\mathsf{sd}_3(\pi_3) = 5\,1\,2\,3\,4$, while $\mathsf{sd}_4(\pi_3) = 3\,4\,5\,1\,2$.

(iv) The simple operations yield a nondeterministic process: there are permutations having both successful and unsuccessful sorting strategies. One such permutation is $\pi_4 = 1\,3\,5\,7\,9\,2\,4\,6\,8$. Note that $\mathsf{sd}_3(\mathsf{sd}_5(\mathsf{sd}_7(\pi_4))) = 1\,9\,2\,3\,4\,5\,6\,7\,8$ is an unsortable permutation. However, $\pi_4$ can be sorted, e.g., by the following strategy: $\mathsf{sd}_2(\mathsf{sd}_4(\mathsf{sd}_6(\mathsf{sd}_8(\pi_4)))) = 1\,2\,3\,4\,5\,6\,7\,8\,9$.

(v) Permutation $\pi_5 = 1\,3\,5\,2\,4$ has both successful and unsuccessful sorting strategies. Indeed, $\mathsf{sd}_3(\pi_5) = 1\,5\,2\,3\,4$, an unsortable permutation. However, $\mathsf{sd}_2(\mathsf{sd}_4(\pi_5)) = 1\,2\,3\,4\,5$ is sorted.

(vi) Applying a cyclic shift to a permutation may render it unsortable. Indeed, permutation $2\,1\,4\,3\,5$ is sortable, while $5\,2\,1\,4\,3$ is not.

(vii) Consider the signed permutation $\pi_7 = 1\,11\,3\,9\,5\,7\,2\,4\,13\,6\,15\,8\,10\,12\,14\,16$. Operation $\mathsf{sd}$ may be applied to $\pi_7$ on integers 3, 6, 9, 11, 13, and 15 . Doing that however leads to a unsortable permutation:

$$\mathsf{sd}_3(\mathsf{sd}_6(\mathsf{sd}_9(\mathsf{sd}_{11}(\mathsf{sd}_{13}(\mathsf{sd}_{15}(\pi_7)))))) = 1\,5\,6\,7\,2\,3\,4\,8\,9\,10\,11\,12\,13\,14\,15\,16.$$

However, omitting $\mathsf{sd}_3$ from the above composition leads to a sorting strategy for $\pi_7$: let

$$\pi_7' = \mathsf{sd}_6(\mathsf{sd}_9(\mathsf{sd}_{11}(\mathsf{sd}_{13}(\mathsf{sd}_{15}(\pi_7))))) = 1\,3\,5\,6\,7\,2\,4\,8\,9\,10\,11\,12\,13\,14\,15\,16.$$

Then $\mathsf{sd}_2(\mathsf{sd}_4(\pi_7'))$ is a sorted permutation.

(viii) Consider the signed permutation $\pi_8$ associated to the actin gene in S.nova, $\pi_8 = 3\,4\,6\,5\,7\,9\,\overline{2}\,1\,8$. A sorting strategy for $\pi_8$ is shown bellow (compare it with Example 3):

$$\begin{aligned}
\mathsf{sd}_5(\pi_8) &= 3\,4\,5\,6\,7\,9\,\overline{2}\,1\,8\\
\mathsf{sd}_8(\mathsf{sd}_5(\pi_8)) &= 3\,4\,5\,6\,7\,8\,9\,\overline{2}\,1\\
\mathsf{sh}_2(\mathsf{sd}_8(\mathsf{sd}_5(\pi_8))) &= \overline{9}\,\overline{8}\,\overline{7}\,\overline{6}\,\overline{5}\,\overline{4}\,\overline{3}\,\overline{2}\,1\\
\mathsf{sh}_1(\mathsf{sh}_2(\mathsf{sd}_8(\mathsf{sd}_5(\pi_8)))) &= \overline{9}\,\overline{8}\,\overline{7}\,\overline{6}\,\overline{5}\,\overline{4}\,\overline{3}\,\overline{2}\,\overline{1}.
\end{aligned}$$

## 4.3 Modelling by signed double occurrence strings

The structure of a gene may be simplified by representing only the sequence of its pointers, 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 $\overline{\Sigma}$ its signed copy. A string $v \in (\Sigma \cup \overline{\Sigma})^*$ is a *signed double occurrence string* if for every letter $a \in \mathsf{dom}(v)$, $v$ 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 $u = 2\,4\,3\,\overline{2}\,\overline{5}\,3\,4\,5$ 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 $w = 2\,4\,3\,\overline{2}\,\overline{5}\,3\,5$ 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: $\mu((i,j)) = i\,j$, for all $2 \leq i < j \leq \kappa$, $\mu((b,k)) = i$, $\mu((k,e)) = i$, for all $2 \leq k \leq \kappa$, and $\mu((b,e)) = \lambda$. Then $\mu$ defines a morphism from the set of MDS descriptors to the set of legal string. We say that $\mu(\delta)$ is the legal string associated to $\delta$.

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

$$3\,4\,4\,5\,6\,7\,5\,6\,7\,8\,9\,\overline{3}\,\overline{2}\,2\,8\,9.$$

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 $\mathsf{ld}$, $\mathsf{sh}$ and $\mathsf{sd}$ 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 $\mathsf{ld}$-*rule* for $p$ is defined by

$$\mathsf{ld}_p(u_1\,p\,p\,u_2) = u_1\,u_2.$$

where $u_1, u_2 \in (\Sigma \cup \overline{\Sigma})^*$. Let $\mathsf{Ld} = \{\mathsf{ld}_p \mid p \in \Pi_\kappa,\ \kappa \geq 2\}$.

(2") For each pointer $p \in \Pi_\kappa$, the $\mathsf{sh}$-*rule* for $p$ is defined by

$$\mathsf{sh}_p(u_1\,p\,u_2\,\overline{p}\,u_3) = u_1\,\overline{u}_2\,u_3,$$

where $u_1, u_2, u_3 \in (\Sigma \cup \overline{\Sigma})^*$ and $|u_2| \leq 1$. Let $\mathsf{Sh} = \{\mathsf{sh}_p \mid p \in \Pi_\kappa,\ \kappa \geq 2\}$.

(3") For each pointers $p, q \in \Pi_\kappa$, the $\mathsf{sd}$-*rule* for $p, q$ is defined by

$$\mathsf{sd}_{p,q}(u_1\,p\,q\,u_2\,p\,q u_3) = u_1\,u_2\,u_3,$$

where $u_1, u_2, u_3 \in (\Sigma \cup \overline{\Sigma})^*$. Let $\mathsf{Sd} = \{\mathsf{sd}_{p,q} \mid p, q \in \Pi_\kappa,\ \kappa \geq 2\}$.

A composition $\varphi = \varphi_n \ldots \varphi_1$ of operations from $\mathsf{Ld} \cup \mathsf{Sh} \cup \mathsf{Sd}$ 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 $u = 3\,4\,4\,5\,6\,7\,5\,6\,7\,8\,9\,\overline{3}\,\overline{2}\,2\,8\,9$. Here is a successful reduction of $u$ using only simple operations (compare it with Examples 3 and 4(viii)):

$$\begin{aligned}
&\mathsf{ld}_4(u) = 3\,5\,6\,7\,5\,6\,7\,8\,9\,\overline{3}\,\overline{2}\,2\,8\,9\\
&\mathsf{sd}_{5,6}(\mathsf{ld}_4(u)) = 3\,7\,7\,8\,9\,\overline{3}\,\overline{2}\,2\,8\,9\\
&\mathsf{ld}_7(\mathsf{sd}_{5,6}(\mathsf{ld}_4(u))) = 3\,8\,9\,\overline{3}\,\overline{2}\,2\,8\,9\\
&\mathsf{sd}_{8,9}(\mathsf{ld}_7(\mathsf{sd}_{5,6}(\mathsf{ld}_4(u)))) = 3\,\overline{3}\,\overline{2}\,2\\
&\mathsf{sh}_3(\mathsf{sd}_{8,9}(\mathsf{ld}_7(\mathsf{sd}_{5,6}(\mathsf{ld}_4(u))))) = \overline{2}\,2\\
&\mathsf{sh}_2(\mathsf{sh}_3(\mathsf{sd}_{8,9}(\mathsf{ld}_7(\mathsf{sd}_{5,6}(\mathsf{ld}_4(u)))))) = \lambda.
\end{aligned}$$

## 5 Discussion

In this paper we introduced a molecular model of the so-called simple operations, a restricted variant of the intramolecular model for gene assembly. In simple operations the type of fold that a micronuclear chromosome has to make during an 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. A number of questions (research topics) considering simple operations are natural and worth investigating. 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 detailed study of simple operations was already initiated in [12].

**Acknowledgments** The authors gratefully acknowledge support from the European Union project MolCoNet, IST-2001-32008, the European Research Training Network SegraVis, Academy of Finland (project 39802 to TH and project 203667 to IP), and NSF (grant 0121422 to GR).

## References

[1] Berman, P., and Hannenhalli, S., Fast sorting by reversals. *Combinatorial Pattern Matching, Lecture Notes in Comput. Sci.* **1072** (1996) 168–185.

[2] Caprara, A., Sorting by reversals is difficult. In S. Istrail, P. Pevzner and M. Waterman (eds.) *Proceedings of the 1st Annual International Conference on Computational Molecular Biology* (1997) pp. 75–83.

[3] Cavalcanti, A., Clarke, T.H., Landweber, L., MDS_IES_DB: a database of macronuclear and micronuclear genes in spirotrichous ciliates. *Nucleic Acids Research* **33** (2005) 396–398.

[4] Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D. M., and Rozenberg, G., Formal systems for gene assembly in ciliates. *Theoret. Comput. Sci.* **292** (2003) 199–219.

[5] Ehrenfeucht, A., Harju, T., Petre, I., and Rozenberg, G., Characterizing the micronuclear gene patterns in ciliates. *Theory of Comput. Syst.* **35** (2002) 501–519.

[6] Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D. M., and Rozenberg, G., *Computation in Living Cells: Gene Assembly in Ciliates*, Springer (2003).

[7] Ehrenfeucht, A., Petre, I., Prescott, D. M., and Rozenberg, G., Universal and simple operations for gene assembly in ciliates. In: V. Mitrana and C. Martin-Vide (eds.) *Words, Sequences, Languages: Where*

*Computer Science, Biology and Linguistics Meet*, Kluwer Academic, Dortrecht, (2001) pp. 329–342.

[8] Ehrenfeucht, A., Petre, I., Prescott, D. M., and Rozenberg, G., String and graph reduction systems for gene assembly in ciliates. *Math. Structures Comput. Sci.* **12** (2001) 113–134.

[9] Ehrenfeucht, A., Petre, I., Prescott, D. M., and Rozenberg, G., Circularity and other invariants of gene assembly in cliates. In: M. Ito, Gh. Păun and S. Yu (eds.) *Words, semigroups, and transductions*, World Scientific, Singapore, (2001) pp. 81–97.

[10] Ehrenfeucht, A., Prescott, D. M., and Rozenberg, G., Computational aspects of gene (un)scrambling in ciliates. In: L. F. Landweber, E. Winfree (eds.) *Evolution as Computation*, Springer, Berlin, Heidelberg, New York (2001) pp. 216–256.

[11] Hannenhalli, S., and Pevzner, P. A., Transforming cabbage into turnip (Polynomial algorithm for sorting signed permutations by reversals). In: *Proceedings of the 27th Annual ACM Symposium on Theory of Computing* (1995) pp. 178–189.

[12] Harju, T., Petre, I., Rogojin, V. and Rozenberg, G., Simple operations for gene assembly. In: *Preproceedings of the 11th International meeting on DNA-based computing*, 2005, to appear.

[13] Harju, T., Petre, I., Li, C. and Rozenberg, G., Parallelism in gene assembly. In: *Proceedings of DNA-based computers 10*, Springer, to appear, 2005.

[14] Harju, T., Petre, I., and Rozenberg, G., Gene assembly in ciliates: molecular operations. In: G.Paun, G. Rozenberg, A.Salomaa (Eds.) *Current Trends in Theoretical Computer Science*, (2004).

[15] Harju, T., Petre, I., and Rozenberg, G., Gene assembly in ciliates: formal frameworks. In: G.Paun, G. Rozenberg, A.Salomaa (Eds.) *Current Trends in Theoretical Computer Science*, (2004).

[16] Jahn, C. L., and Klobutcher, L. A., Genome remodeilng in ciliated protozoa. *Ann. Rev. Microbiol.* **56** (2000), 489–520.

[17] Kaplan, H., Shamir, R., and Tarjan, R. E., A faster and simpler algorithm for sorting signed permutations by reversals. *SIAM J. Comput.* **29** (1999) 880–892.

[18] Kari, L., and Landweber, L. F., Computational power of gene rearrangement. In: E. Winfree and D. K. Gifford (eds.) *Proceedings of DNA Bases Computers, V* American Mathematical Society (1999) pp. 207–216.

[19] Landweber, L. F., and Kari, L., The evolution of cellular computing: Nature's solution to a computational problem. In: *Proceedings of the 4th DIMACS Meeting on DNA-Based Computers*, Philadelphia, PA (1998) pp. 3–15.

[20] Landweber, L. F., and Kari, L., Universal molecular computation in ciliates. In: L. F. Landweber and E. Winfree (eds.) *Evolution as Computation*, Springer, Berlin Heidelberg New York (2002).

[21] Prescott, D. M., *Cells: Principles of Molecular Structure and Function*, Jones and Barlett, Boston (1988).

[22] Prescott, D. M., Cutting, splicing, reordering, and elimination of DNA sequences in hypotrichous ciliates. *BioEssays* **14** (1992) 317–324.

[23] Prescott, D. M., The unusual organization and processing of genomic DNA in hypotrichous ciliates. *Trends in Genet.* **8** (1992) 439–445.

[24] Prescott, D. M., The DNA of ciliated protozoa. *Microbiol. Rev.* **58**(2) (1994) 233–267.

[25] Prescott, D. M., The evolutionary scrambling and developmental unscabling of germlike genes in hypotrichous ciliates. *Nucl. Acids Res.* **27** (1999), 1243 – 1250.

[26] Prescott, D. M., Genome gymnastics: unique modes of DNA evolution and processing in ciliates. *Nat. Rev. Genet.* 1(3) (2000) 191–198.

[27] Prescott, D. M., and DuBois, M., Internal eliminated segments (IESs) of Oxytrichidae. *J. Eukariot. Microbiol.* **43** (1996) 432–441.

[28] Prescott, D. M., Ehrenfeucht, A., and Rozenberg, G., Molecular operations for DNA processing in hypotrichous ciliates. *Europ. J. Protistology* **37** (2001) 241–260.

[29] Prescott, D. M., and Rozenberg, G., How ciliates manipulate their own DNA – A splendid example of natural computing. *Natural Computing* **1** (2002) 165–183.

[30] Prescott, D. M., and Rozenberg, G., Encrypted genes and their reassembly in ciliates. In: M. Amos (ed.) *Cellular Computing*, Oxford University Press, Oxford (2003).

University of Turku
- Department of Information Technology
- Department of Mathematics

Åbo Akademi University
- Department of Computer Science
- Institute for Advanced Management Systems Research

Turku School of Economics and Business Administration
- Institute of Information Systems Sciences

ISBN 952-12-1579-8
ISSN 1239-1891