Simple Operations for Gene Assembly

Tero Harju[1,4], Ion Petre[2,3,4],
Vladimir Rogojin[3,4], and Grzegorz Rozenberg[5,6]

[1] Department of Mathematics, University of Turku,
Turku 20014, Finland
harju@utu.fi
[2] Academy of Finland
[3] Department of Computer Science, Åbo Akademi University,
Turku 20520, Finland
ipetre@abo.fi, vrogojin@abo.fi
[4] Turku Centre for Computer Science,
Turku 20520, Finland
[5] Leiden Institute for Advanced Computer Science, Leiden University,
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
[6] Department of Computer Science, University of Colorado at Boulder,
Boulder, Co 80309-0347, USA
rozenber@liacs.nl

Abstract. The intramolecular model for gene assembly in ciliates considers three operations, ld, hi, and dlad that can assemble any 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 consist of many coding and non-coding blocks. We consider in this paper some simpler variants of the three operations, where only one coding block is rearranged at a time. We characterize in this paper the gene patterns that can be assembled through these variants. Our characterization is in terms of signed permutations and dependency graphs. 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 assembling strategy.

1 Introduction

The 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 same 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 is shuffled, with some of the MDSs being inverted. The structure is particularly complex in a family of ciliates called *Stichotrichs* – we concentrate in this paper on this family. During the process of sexual reproduction,

A. Carbone and N.A. Pierce (Eds.): DNA11, LNCS 3892, pp. 96–111, 2006.

ciliates destroy the old macronuclei and transform a micronucleus into a new macronucleus. In this process, ciliates must assemble all genes by placing in the orthodox order all MDSs. 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 [9] and [27] consists of three operations: ld, hi, and dlad. In each of these operations, the molecule 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 [15], [20], [21], [22], [23], [24], [25], [26] and for details related to the intramolecular model and its mathematical formalizations we refer to [3], [4], [7], [8], [11], [12], [13], [28], [29], as well as to the recent monograph [5]. For a different intermolecular model we refer to [17], [18], [19].

In general there are no restrictions on the number of nucleotides between the two pointers that should be aligned in a certain fold. However, all available experimental data is consistent with restricted versions of our operations, in which between two aligned pointers there is never more than one MDS, see [5] and [6]. We propose in this paper a mathematical model for simple variants of ld, hi, and dlad. The model, in terms of signed permutations, is used to answer the following question: which gene patterns can be assembled by the simple operations? As it turns out, the question is difficult: the simple assembly is a non-deterministic process, with more than one strategy possible for certain patterns and in some cases, with both successful and unsuccessful assemblies. We completely answer the question in terms of sorting signed permutations. Here, a signed permutation represents the sequence of MDSs in a gene pattern, including their orientation.

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], [10], [16].

2 Mathematical Preliminaries

For an alphabet Σ we denote by Σ^* the set of all finite strings over Σ. For a string u we denote $\mathsf{dom}(u)$ the set of letters occurring in u. We denote by Λ the empty string. For strings u, v over Σ, we say that u is a *substring* of v, denoted $u \leq v$, if $v = xuy$, for some strings x, y. We say that u is a *subsequence* of v, denoted $u \leq_s v$, if $u = a_1a_2 \dots a_m$, $a_i \in \Sigma$ and $v = v_0a_1v_1a_2 \dots a_mv_m$, for some strings v_i, $0 \leq i \leq m$, over Σ. For some $A \subseteq \Sigma$ we define the morphism $\phi_A : \Sigma^* \to A^*$ as follows: $\phi_A(a_i) = a_i$, if $a_i \in A$ and $\phi_A(a_i) = \Lambda$ if $a_i \in \Sigma \setminus A$. For any $u \in \Sigma^*$, we denote $u|_A = \phi_A(u)$. We say that the *relative positions* of letters from set $A \subseteq \Sigma$ are the same in strings $u, v \in \Sigma^*$ if and only if $u|_A = v|_A$.

Let $\Sigma_n = \{1, 2, \ldots, n\}$ and let $\overline{\Sigma}_n = \{\overline{1}, \overline{2}, \ldots, \overline{n}\}$ be a *signed copy* of Σ_n. For any $i \in \Sigma_n$ we say that i is a *unsigned letter*, while $\overline{i}$ is a *signed* letter. Let $\|.\|$ be the morphism from $(\Sigma_n \cup \overline{\Sigma}_n)^*$ to Σ_n^* that unsigns the letters: for all $a \in \Sigma_n$, $\|\overline{a}\| = \|a\| = a$. For a string u over $\Sigma_n \cup \overline{\Sigma}_n$, $u = a_1 a_2 \ldots a_m$, $a_i \in \Sigma_n \cup \overline{\Sigma}_n$, for all $1 \leq i \leq m$, we denote its *inversion* by $\overline{u} = \overline{a}_m \ldots \overline{a}_2 \overline{a}_1$, where $\overline{\overline{a}} = a$, for all $a \in \Sigma_n$.

Consider a *bijective mapping* (called *permutation*) $\pi : \Delta \to \Delta$ over an alphabet $\Delta = \{a_1, a_2, \ldots, a_l\}$ with the order relation $a_i \leq a_j$ for all $i \leq j$. We often identify π with the string $\pi(a_1)\pi(a_2)\ldots\pi(a_l)$. The domain of π, denoted $\mathsf{dom}(\pi)$, is Δ. We say that π is *(cyclically) sorted* if $\pi = a_k\, a_{k+1} \ldots a_l\, a_1\, a_2 \ldots a_{k-1}$, for some $1 \leq k \leq l$.

A *signed permutation* over Δ is a string ψ over $\Delta \cup \overline{\Delta}$ such that $\|\psi\|$ is a permutation over Δ. We say that ψ is *(cyclically) sorted* if $\psi = a_k\, a_{k+1} \ldots a_l\, a_1\, a_2 \ldots a_{k-1}$ or $\psi = \overline{a}_{k-1} \ldots \overline{a}_2\, \overline{a}_1\, \overline{a}_l \ldots \overline{a}_{k+1}\, \overline{a}_k$, for some $1 \leq k \leq l$. Equivalently, ψ is sorted if either ψ, or $\overline{\psi}$ is a sorted unsigned permutation. In the former case we say that ψ is sorted in the *orthodox order*, while in the latter case we say that ψ is sorted in the *inverted order*.

3 The Intramolecular Model

Three molecular operations, ld, hi, dlad were conjectured in [9] and [27] for gene assembly. We only show here the folding and the recombinations taking place in each case, referring for more details to [5]. 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.

3.1 Simple Operations

Note that all three operations ld, hi, dlad are *intramolecular*, that is, a single molecule folds on itself to rearrange its coding blocks. Thus, since ld excises one circular molecule, that circular molecule can only contain noncoding blocks (or, in a special case, contain the entire gene, see [5] 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 combines two consecutive MDSs into a bigger composite MDS. E.g., consider that $M_i M_{i+1}$ 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. Then ld makes a fold as in Fig. 1 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.

In the case of hi and dlad, the rearranged sequences may be arbitrarily large. E.g., the actin I gene in S.nova has the following sequence of MDSs: $M_3 M_4 M_6 M_5 M_7 M_9 \overline{M}_2 M_1 M_8$, where MDS M_2 is inverted. Here, 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

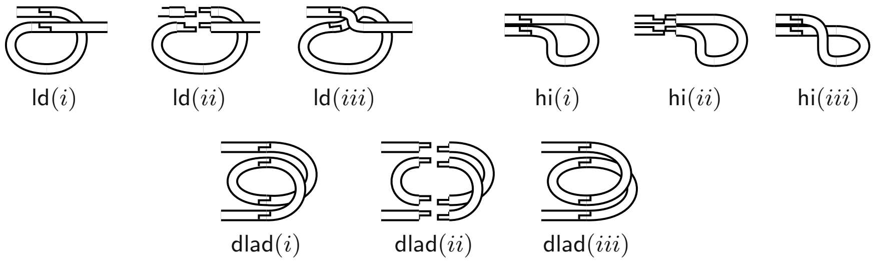

Fig. 1. Illustration of the ld, hi, dlad molecular operation showing in each case: (i) the folding, (ii) the recombination, and (iii) the result

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).

As it turns out, all available experimental data is consistent with applications of 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.

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 Fig. 2, where the MDSs are indicated by rectangles and their flanking pointers are shown.

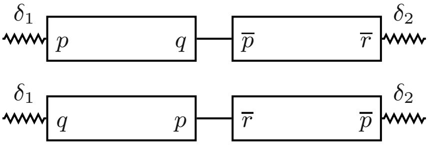

Fig. 2. The MDS/IES structures where the *simple hi*-rule is applicable. Between the two MDSs there is only one IES.

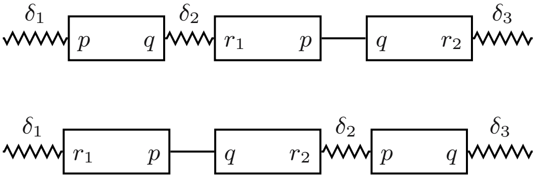

Fig. 3. The MDS/IES structures where the *simple dlad*-rule is applicable. Straight line denotes one IES.

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 Fig. 3.

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 Gene Assembly as a Sorting of Signed Permutations

The gene structure of a ciliate can 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 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.

When formalizing the gene assembly as a sorting of permutations we 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 [3] and [7] 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.

The simple hi is formalized on permutations through operation sh. For each $p \geq 1$, sh_p is defined as follows:

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

where x, y are signed strings over Σ_n. We denote $\mathsf{Sh} = \{\mathsf{sh}_i \mid 1 \leq i \leq n\}$.

The simple dlad is formalized on permutations through operation sd. For each p, $2 \leq p \leq n-1$, sd_p is defined as follows:

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

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

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

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

We say that a signed permutation π over the set of integers Σ_n 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 sorted permutation. In this case $\Phi = \phi_1 \circ \ldots \circ \phi_k$ is a *sorting strategy* for π. Permutation π is Sh-*sortable* if $\phi_1, \ldots, \phi_k \in \mathsf{Sh}$ and π is Sd-*sortable* if $\phi_1, \ldots, \phi_k \in \mathsf{Sd}$. We say that ϕ_i is *part* of Φ and also that ϕ_i is used in Φ before ϕ_j for all $1 \leq j < i \leq k$.

Example 1. (i) Permutation $\pi_1 = 3\,\overline{4}\,\overline{5}\,6\,\overline{1}\,2$ is sortable and a sorting strategy is $\mathsf{sh}_1(\mathsf{sh}_5(\mathsf{sh}_4(\pi_1))) = 3\,4\,5\,6\,1\,2$. Permutation $\pi_1' = 3\,4\,5\,6\,\overline{1}\,\overline{2}$ is unsortable. Indeed, no sh operations and no sd operation is applicable to π_1'.

(ii) Permutation $\pi_2 = 1\,3\,4\,2\,\overline{5}$ is sortable and has only one sorting strategy: $\mathsf{sh}_5(\mathsf{sd}_2(\pi_2)) = 1\,2\,3\,4\,5$.

(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, $\mathsf{sd}_3(\pi_3) = 5\,1\,2\,3\,4$. At the same time, $\mathsf{sd}_4(\pi_3) = 3\,4\,5\,1\,2$.

(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 a unsortable permutation. However, π_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, $\mathsf{sd}_3(\pi_5) = 1\,5\,2\,3\,4$, a 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 sd may be applied to π_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 sd_3 from the above composition leads to a sorting strategy for π_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.

Lemma 1. *Let π be a signed permutation over Σ_n and $i \in \Sigma_n \cup \overline{\Sigma_n}$. Then we have the following properties:*

(i) If sd_i is applicable to π, then $\mathsf{sd}_{\overline{i}}$ is applicable to $\overline{\pi}$ and $\overline{\mathsf{sd}_i(\pi)} = \mathsf{sd}_{\overline{i}}(\overline{\pi})$.

(ii) If sh_i, where i is unsigned, is applicable to π, then sh_{i-1} or sh_{i+1} is applicable to $\overline{\pi}$ and $\overline{\mathsf{sh}_i(\pi)} = \mathsf{sh}_{i-1}(\overline{\pi})$ or $\overline{\mathsf{sh}_i(\pi)} = \mathsf{sh}_{i+1}(\overline{\pi})$.

5 Sh-Sortable Permutations

We characterize in this section all signed permutations that can be sorted using only the Sh operations. As it turns out, their form is easy to describe since the Sh operations do not change the relative positions of the letters in the permutation.

The following result characterizes all Sh-sortable signed permutations.

Theorem 1. *A signed permutation $\pi = p_1 \dots p_n$, $p_i \in \ \Sigma_n \cup \overline{\Sigma_n}$, is sh-sortable if and only if*

(i) $\|\pi\| = k\,(k+1)\dots n\,1\dots(k-1)$, for some $1 \leq k \leq n$ and there are i, j, $1 \leq i \leq k-1$, $k \leq j \leq n$ such that p_i and p_j are unsigned letters, or
(ii) $\|\pi\| = (k-1)\dots 1\,n\dots(k+1)\,k$, for some $1 \leq k \leq n$ and there are i, j, $1 \leq i \leq k-1$, $k \leq j \leq n$ such that p_i and p_j are signed letters.

In Case (i), π sorts to $k\,(k+1)\dots n\,1\dots(k-1)$, while in Case (ii), π sorts to $(\overline{k-1})\dots\overline{1}\,\overline{n}\dots(\overline{k+1})\,\overline{k}$.

Example 2. (i) Permutation $\pi_1 = \overline{5}\,\overline{6}\,7\overline{8}\,\overline{1}\,2\overline{3}\,\overline{4}$ is Sh sortable and an Sh-sorting for π_1 is $\mathsf{sh}_4(\mathsf{sh}_3(\mathsf{sh}_1(\mathsf{sh}_8(\mathsf{sh}_5(\mathsf{sh}_6(\pi_1)))))) = 5\,6\,7\,8\,1\,2\,3\,4$. Note that sh_5 can be applied only after sh_6 and also, sh_4 can be applied only after sh_3.
(ii) Permutation $\pi_2 = \overline{5}\,\overline{6}\,7\overline{8}\,\overline{1}\,\overline{2}\,\overline{3}\,\overline{4}$ is unsortable, since we cannot unsign 1, 2, 3 and 4.

6 Sd-Sortable Permutations

We characterize in this section the Sd-sortable permutations. A crucial role in our result is played by the dependency graph of a signed permutation.

6.1 The Dependency Graph

This is in general a directed graph with self-loops: there may be edges from a node to itself. The dependency graph describes for a permutation π the order in which Sd-operations can be applied to π.

For a permutation π over Σ_n we define its dependency graph as the directed graph $G_\pi = (\Sigma_n, E)$, where $(i, j) \in E$, $1 \leq i \leq n$, $2 \leq j \leq n-1$, if and only if $(j-1)i(j+1) \leq_s \pi$. Also, if $(j+1)(j-1) \leq_s \pi$, then $(j, j) \in E$. Intuitively, the edge (i, j) represents that the rule sd_j may be applied in a sorting strategy for π only after rule sd_i has been applied. A loop (i, i) represents that sd_i can never be used in a sorting strategy for π. Note that G_π may also have a loop on node i if $(i-1)i(i+1) \leq_s \pi$.

Example 3. (i) The graph associated to permutation $\pi_1 = 1\,4\,3\,6\,5\,7\,2$ is shown in Fig. 4(a). It can be seen, e.g., that sd_3 can never be applied in a sorting strategy for π and because of edge $(3, 5)$, neither can sd_5. Also, the graph suggests that sd_6 should be applied before sd_4 and this one before sd_2. Indeed, $\mathsf{sd}_2(\mathsf{sd}_4(\mathsf{sd}_6(\pi))) = 1\,2\,3\,4\,5\,6\,7$.
(ii) The graph associated to permutation $\pi_2 = 1\,4\,3\,2\,5$ is shown in Fig. 4(b). Thus, the graph has a cycle with nodes 2 and 4. Indeed, to apply sd_2 in a strategy for π_2, sd_4 should be applied first and the other way around.

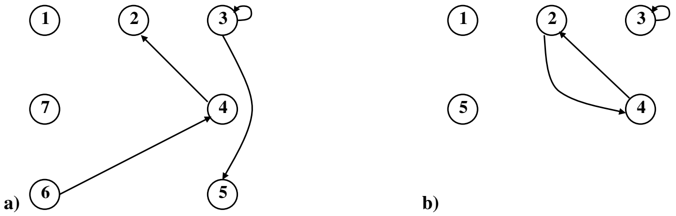

Fig. 4. Dependency graphs (a) associated to $\pi_1 = 1\,4\,3\,6\,5\,7\,2$ and (b) associated to $\pi_2 = 1\,4\,3\,2\,5$

Lemma 2. *Let π be a unsigned permutation over Σ_n and $G_\pi = (\Sigma_n, E)$ its dependency graph.*

(i) There exists no sorting strategy Φ for π such that sd_i and sd_{i+1} are both used in Φ, for some $1 \leq i \leq n-1$.

(ii) If sd_j is used in a sorting strategy for π and $(i,j) \in E$, for some $i, j \in \Sigma_n$, then sd_i is also used, before sd_j, in the same sorting strategy.

(iii) If there is a path from i to j in G_π, then in any strategy where sd_j is used, sd_i is also used, before sd_j.

(iv) If G_π has a cycle containing $i \in \Sigma_n$, then sd_i cannot be applied in any sorting strategy of π.

(v) There is no strategy where sd_1 and sd_n can be applied.

6.2 The Characterization

We characterize in this subsection the Sd-sortable permutations. We first give an example.

Example 4. Consider the dependency graph G_π for $\pi = 1\,11\,3\,9\,5\,7\,2\,4\,13\,6\,15\,8\,10\,12\,14\,16$, shown in Fig. 5. Based on Lemma 2 and G_π we build a sorting strategy Φ for π. We label all nodes i for which sd_i is used in Φ by M and the other nodes by U. Nodes labelled by M are shown with a white background in Fig. 5, while nodes labelled by U are shown with black one.

By Lemma 2(iv)(v) operations sd_1, sd_8, sd_{10} and sd_{16} cannot be applied in any strategy of π. Thus, $1, 8, 10, 16 \in U$. Now, to apply operation sd_2, since we have edge $(11, 2)$ in the dependency graph G_π, it follows by Lemma 2(ii) that sd_{11} must be applied in the same strategy as sd_2. Thus, $2, 11 \in M$. According to Lemma 2(i) we cannot apply sd_2 and sd_3 in the same strategy, thus we label 3 by U. To use sd_4, since edge $(9, 4)$ is present in the dependency graph, we need to label both 4 and 9 by M. It follows then by Lemma 2(i) that $5 \in U$. Then 6 can be labelled by M and then, necessarily, $7 \in U$. Note now, that if $12 \in M$, since $(3, 12)$ is an edge in G_π, then by Lemma 2(ii), $3 \in M$, which contradicts our labelling of 3. Thus, $12 \in U$. Then 13 can be labelled by M and necessarily, $14 \in U$. Also, 15 can now be labelled by M.

In this way we obtain $M = \{2, 4, 6, 9, 11, 13, 15\}$ and $U = \{1, 3, 5, 7, 8, 10, 12, 14, 16\}$. Note that, since elements in U do not change their relative positions in the strategy Φ we are building, $\pi|_U$ has to be sorted: $\pi|_U = 1\,3\,5\,7\,8\,10\,12\,14\,16$.

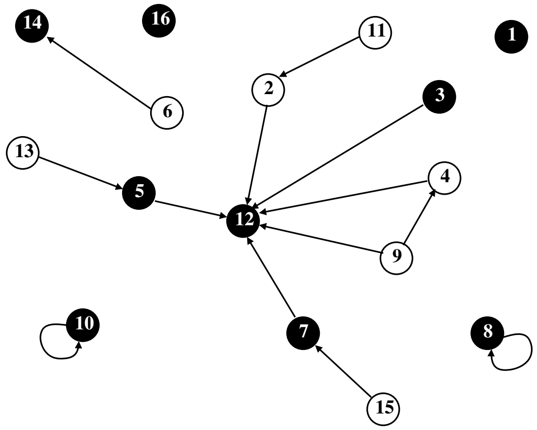

Fig. 5. The dependency graph associated to $\pi = 1\,11\,3\,9\,5\,7\,2\,4\,13\,6\,15\,8\,10\,12\,14\,16$

Our strategy Φ is now a composition of operations sd_i, with $i \in M$. The dependency graph shows the order in which these operations must be applied, i.e., sd_2 can be applied only after sd_{11} and sd_4 can be applied only after sd_9. In this way, we can sort π by applying the following sorting strategy:

$$(\mathsf{sd}_2 \circ \mathsf{sd}_4 \circ \mathsf{sd}_7 \circ \mathsf{sd}_{15} \circ \mathsf{sd}_{13} \circ \mathsf{sd}_{11} \circ \mathsf{sd}_9)(\pi) = 1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14\,15\,16.$$

Clearly, our choice of M and U is not unique. For instance, we may have $M = \{2, 4, 7, 9, 11, 13, 15\}$ and $U = \{1, 3, 5, 6, 8, 10, 12, 14, 16\}$ as shown in Fig. 6. The strategy will be in this case $\mathsf{sd}_2 \circ \mathsf{sd}_4 \circ \mathsf{sd}_6 \circ \mathsf{sd}_{15} \circ \mathsf{sd}_{13} \circ \mathsf{sd}_{11} \circ \mathsf{sd}_9$.

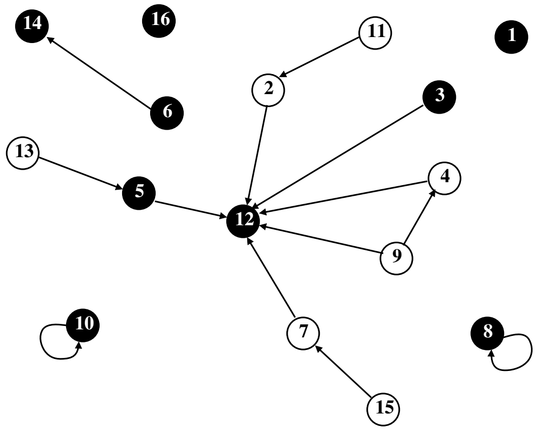

Fig. 6. The dependency graph associated to $\pi = 1\,11\,3\,9\,5\,7\,2\,4\,13\,6\,15\,8\,10\,12\,14\,16$

The following result characterizes all Sd-sortable permutations.

Theorem 2. *Let π be a unsigned permutation. Then π is* Sd*-sortable if and only if there exists a partition $\{1, 2, \ldots, n\} = M \cup U$, such that the following conditions are satisfied:*

(i) $\pi|_U$ is sorted;
(ii) Nodes of M induce an acyclic dependency subgraph;
(iii) If $k \to l$ is a dependency of π and $l \in M$, then $k \in M$;
(iv) For any $k \in M$, $(k-1)(k+1) \leq_s \pi$;
(v) For any $k \in M$, $(k-1), (k+1) \in U$.

Example 5. Consider permutation $\pi = 1\,3\,8\,10\,5\,7\,2\,9\,11\,4\,6\,12$. Its dependency graph is shown in Fig. 7. Based only on this graph and using Theorem 2 we deduce a sorting strategy for π.

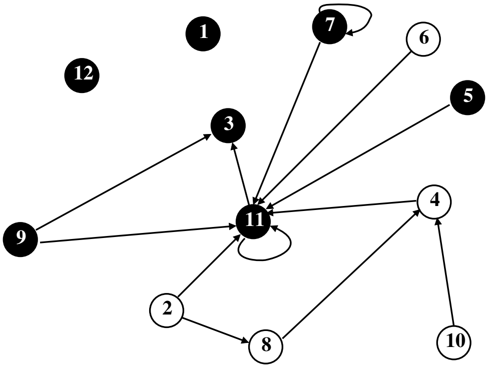

Fig. 7. The dependency graph associated to $\pi = 1\,3\,8\,10\,5\,7\,2\,9\,11\,4\,6\,12$

It follows from Theorem 2(ii) that $7, 11 \in U$. Then it follows from Theorem 2(iii) that $3 \in U$ and from Theorem 2(v) that $1, 12 \in U$. Since $1, 3 \in U$, it follows from Theorem 2(i) that $2 \in M$. Also, since $3, 7, 11 \in U$, it follows from Theorem 2(i) that $4, 6, 8, 10 \in M$ and so, by Theorem 2(v), $5, 9 \in U$. We have now a complete labelling of G_π:

$$M = \{2, 4, 6, 8, 10\},\ U = \{1, 3, 5, 7, 9, 11, 12\}$$

Permutation π may be sorted now by a composition of operations sd_i with $i \in M$.

The dependency graph imposes the following order of operations: sd_4 after sd_8 and sd_{10}, sd_8 after sd_2. The other operations can be applied in any order. For instance, we can sort π in the following way:

$$(\mathsf{sd}_4 \circ \mathsf{sd}_8 \circ \mathsf{sd}_2 \circ \mathsf{sd}_{10} \circ \mathsf{sd}_6)(\pi) = 1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12,$$

but also,

$$(\mathsf{sd}_6 \circ \mathsf{sd}_4 \circ \mathsf{sd}_8 \circ \mathsf{sd}_2 \circ \mathsf{sd}_{10})(\pi) == 1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12.$$

7 $\{\mathsf{Sd}, \mathsf{Sh}\}$-Sortable Permutations

We characterize in this section all signed permutations that can be sorted using our operations. First we give some examples.

Example 6. (i) Signed permutations $\pi_1 = 2\,1\,4\,\overline{3}\,5$ and $\pi_2 = 1\,5\,\overline{2}\,4\,3\,6$ are not $\{\mathsf{Sd}, \mathsf{Sh}\}$-sortable. Indeed, just sh_3 can be applied to π_1, but it does not sort it, and no operation can be applied to π_2.

(ii) Signed permutations $\pi_3 = 9\,2\,\overline{10}\,\overline{11}\,1\,5\,3\,7\,\overline{4}\,6\,8$ and $\pi_4 = 5\,4\,\overline{3}\,\overline{8}\,2\,1\,\overline{9}\,7\,\overline{6}$ are $\{\mathsf{Sd}, \mathsf{Sh}\}$-sortable:

$$(\mathsf{sh}_{11} \circ \mathsf{sh}_{10} \circ \mathsf{sd}_2 \circ \mathsf{sd}_5 \circ \mathsf{sh}_4 \circ \mathsf{sd}_7)(\pi_3) = 9\,10\,11\,1\,2\,3\,4\,5\,6\,7\,8$$

and

$$(\mathsf{sh}_4 \circ \mathsf{sh}_2 \circ \mathsf{sh}_3 \circ \mathsf{sh}_3 \circ \mathsf{sd}_{\overline{8}} \circ \mathsf{sh}_6)(\pi_4) = \overline{5}\,\overline{4}\,\overline{3}\,\overline{2}\,\overline{1}\,\overline{9}\,\overline{8}\,\overline{7}\,\overline{6}.$$

Theorem 3. *No permutation π can be sorted both to an orthodox permutation and to an inverted one.*

The following result gives a duality property of sorting signed permutations.

Lemma 3. *A signed permutation π is sortable to an orthodox permutation π_o if and only if its inversion $\overline{\pi}$ is sortable to the inverted permutation $\overline{\pi_o}$.*

The following result is an immediate consequence of Theorem 3 and of Lemma 3.

Corollary 1. *A permutation π is sortable if and only if either π or $\overline{\pi}$ is sortable to an orthodox permutation.*

Consider in the following only permutations π that are sortable to an orthodox form. Let H be the set of all signed letters in π and let Φ_H be a composition of sh-operations applied on all integers in H. Let $D \subseteq \{1, 2, \ldots, n\} \setminus H$ and Φ_D a composition of sd-operations applied on all integers in D. The *dependency graph* $\Gamma_{\pi,\Phi_H,\Phi_D}$ (or just Γ_{Φ_H,Φ_D} when there is no risk of confusion) generated by Φ_H, Φ_D is the following:

- If $j \in D$ (sd_j is in Φ_D) and $(j-1)i(j+1) \leq_s \|\pi\|$, then edge $(\|i\|, j) \in \Gamma_{\Phi_H,\Phi_D}$. Also, if $(j-1) \in H$, then edge $(j-1, j) \in \Gamma_{\Phi_H,\Phi_D}$ and if $j+1 \in H$, then edge $(j+1, j) \in \Gamma_{\Phi_H,\Phi_D}$.
- If $i \in H$ (sh_i is in Φ_H), then we have the following two cases:
  - If sh_i is of the form $(i-1)\overline{i} \rightarrow (i-1)i$, then $(i-1)i \leq_s \|\pi\|$. For any j, if $(i-1)ji \leq_s \|\pi\|$, then $(\|j\|, i) \in \Gamma_{\Phi_H,\Phi_D}$;
  - If sh_i is of the form $\overline{i}(i+1) \rightarrow i(i+1)$, then $i(i+1) \leq_s \|\pi\|$. For any j, if $ij(i+1) \leq_s \|\pi\|$, then $(\|j\|, i) \in \Gamma_{\Phi_H,\Phi_D}$.

Example 7. Consider $\pi = 6\,8\,10\,1\,9\,\overline{3}\,7\,4\,2\,\overline{5}$. Clearly, $H = \{3, 5\}$. Assume we apply Sd operations on 2, 7 and 9, thus $D = \{2, 7, 9\}$. Let us build the dependency graph $G = \Gamma_{\pi,\Phi_H,\Phi_D}$, shown in Fig. 8.

We mark by dashed the nodes in H, by white the nodes in D and we mark by black the rest of vertices. For each vertex i from G we have the following edges (j, i):

- Node 1: we do not have edges $(j, 1)$, since $1 \notin H$ and $1 \notin D$;
- Node 2: $2 \in D$, $3 \in H$, thus $(3, 2) \in G$. Since $1\,9\,3 \leq_s \pi$, we have also $(9, 2) \in G$;
- Node 3: $3 \in H$, $3\,7\,4 \leq_s \|\pi\|$, thus $(7, 3) \in G$;
- Node 4: $4 \notin H$ and $4 \notin D$, thus we have no edges $(j, 4)$;
- Node 5: $5 \in H$ and $4\,2\,5 \leq_s \|\pi\|$, thus $(2, 5) \in G$;
- Node 6: $6 \notin H$ and $6 \notin D$, thus we have no edges $(j, 6)$;
- Node 7: $7 \in D$, $6\,8 \leq \pi$, thus we have no edges $(j, 7)$;
- Node 8: $8 \notin H$ and $8 \notin D$, thus we have no edges $(j, 8)$;
- Node 9: $9 \in D$, $8\,10 \leq \pi$, thus we have no edges $(j, 9)$;
- Node 10: $10 \notin H$ and $10 \notin D$, thus we have no edges $(j, 10)$.

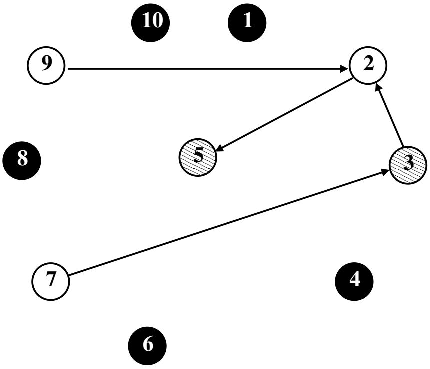

Fig. 8. The dependency graph associated to $\pi = 6\,8\,10\,1\,9\,\overline{3}\,7\,4\,2\,\overline{5}$

Lemma 4. *Let π be an $\mathsf{Sh} \cup \mathsf{Sd}$-sortable permutation over Σ_n and Φ a sorting strategy for π. Let Γ_Φ be the dependency graph associated to π and Φ. Let $\phi_i = \mathsf{sd}_i$ if i is unsigned in π and $\phi_i = \mathsf{sh}_i$ if i is signed in π, for $i \in \Sigma_n$. Then we have the following properties:*

(i) If there is a path from i to j in Γ_Φ and Φ_j is used in Φ, then ϕ_i is applied before ϕ_j in strategy Φ.
(ii) The dependency graph Γ_Φ is acyclic.

The following theorem gives the main result of this section.

Theorem 4. *A permutation π is $\{\mathsf{Sh}, \mathsf{Sd}\}$-sortable to an orthodox form if and only if there is a partition $\{1, 2, \ldots, n\} = D \cup H \cup U$ such that the following conditions are satisfied:*

(i) H is the set of all signed letters in π;
(ii) H sorts $\pi\,|_{H \cup U}$ to an orthodox form with a strategy Φ_H;
(iii) D sorts $\|\pi\|$ with a strategy Φ_D;
(iv) The subgraph of Γ_{Φ_H, Φ_D} induced by $H \cup D$ is acyclic.

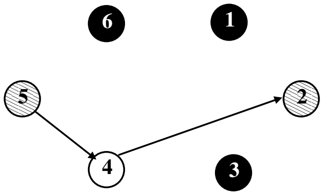

Fig. 9. The dependency graph associated to $\pi = \overline{2}\,4\,3\,\overline{5}\,6\,1$

Example 8. Let $\pi = \overline{2}\,4\,3\,\overline{5}\,6\,1$. We build a sorting strategy for π based on Theorem 4. Consider $H = \{2, 5\}$. Clearly, $\|\pi\| = 2\,4\,3\,5\,6\,1$ is sorted by applying sd_4. Then let $D = \{4\}$ and $U = \{1, 3, 6\}$. We verify now conditions of Theorem 4. Consider $\pi\ |_{H\cup U} = \overline{2}\,3\overline{5}\,6\,1$. Then $\mathsf{sh}_2(\mathsf{sh}_5(\pi\ |_{H\cup U})) = 2\,3\,5\,6\,1$, a (circularly) sorted string. The graph $\Gamma_{\mathsf{sh}_2 \circ \mathsf{sh}_5, \mathsf{sd}_4}$ is shown in Fig. 9, where nodes in H are marked by dashed, nodes in D are marked by white and nodes in U are marked by black. Clearly, $H \cup D$ induces an acyclic subgraph in $\Gamma_{\mathsf{sh}_2 \circ \mathsf{sh}_5, \mathsf{sd}_4}$. Thus, by Theorem 4, π is sortable and a sorting strategy should be obtained by combining $\mathsf{sh}_2 \circ \mathsf{sh}_5$ and sd_4 as indicated by the graph. Since $(4, 2)$ is an edge in the graph, it follows that sd_4 must be applied before sh_2. Also, since $(5, 4)$ is an edge, it follows that sh_5 must be applied before sd_4. Consequently, $\mathsf{sh}_2 \circ \mathsf{sd}_4 \circ \mathsf{sh}_5$ must be a sorting strategy for π. Indeed, $\mathsf{sh}_2(\mathsf{sd}_4(\mathsf{sh}_5(\pi))) = 2\,3\,4\,5\,6\,1$, a (circularly) sorted permutation.

Example 9. Let $\pi = 2\,1\,4\,3\,7\,\overline{5}\,9\,6\,8\,\overline{10}\,11$. We build a sorting strategy for π based on Theorem 4. Clearly, $H = \{5, 10\}$. The unsigned permutation $\|\pi\| = 2\,1\,4\,3\,7\,5\,9\,6\,8\,10\,11$ can be sorted by $\mathsf{sd}_2 \circ \mathsf{sd}_4 \circ \mathsf{sd}_9 \circ \mathsf{sd}_7$, thus $D = \{2, 4, 7, 9\}$. Set $U = \{1, 3, 6, 8, 11\}$. The dependency graph G associated to π and $H \cup U$ is shown in Fig. 10. Clearly, permutation $\pi|_{H\cup U} = 1\,3\,\overline{5}\,6\,8\,\overline{10}\,11$ can be sorted to cyclically sorted permutation $1\,3\,5\,6\,8\,10\,11$ by applying sh_5 and sh_{10}. Also, $H \cup D$

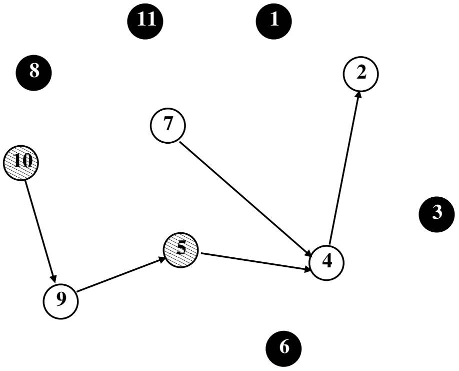

Fig. 10. The dependency graph associated to $\pi = 2\,1\,4\,3\,7\,\overline{5}\,9\,6\,8\,\overline{10}\,11$

induces an acyclic subgraph in G. It follows then that π is sortable. Indeed, a sorting strategy, as suggested by G, is $\mathsf{sd}_2 \circ \mathsf{sd}_4 \circ \mathsf{sd}_7 \circ \mathsf{sh}_5 \circ \mathsf{sd}_9 \circ \mathsf{sh}_{10}$. Another sorting strategy is $\mathsf{sd}_2 \circ \mathsf{sd}_4 \circ \mathsf{sh}_5 \circ \mathsf{sd}_9 \circ \mathsf{sd}_7 \circ \mathsf{sh}_{10}$.

8 Discussion

We consider in this paper a mathematical model for the so called *simple operations* for gene assembly in ciliates. The model we consider here is in terms of *signed permutations*, but the model can also be expressed in terms of *signed double-occurrence strings*, see [14].

Modelling in terms of signed permutations is possible by ignoring the molecular operation Ld that combines two consecutive gene blocks into a bigger block. In this way, the process of combining the sequence of successive coding blocks into one assembled gene becomes the process of sorting the initial sequence of blocks.

It is important to note now that in the molecular model we discus in this paper, each operation affects one single gene block that gets incorporated into a bigger block together with one (in case of Sh) or two (in case of Sd) other blocks. In our mathematical model however, a gene block that was already assembled from several initial blocks is represented as a sorted substring. For that reason, although the molecular operations only displace one block, our model should allow the moving of longer sorted substrings. A mathematical theory in this sense looks challenging. We consider in this paper the simplified variant where our formal operations can only move one block (one letter of the alphabet) at a time. Note however that the general case may in fact be reduced to this simpler variant in the following way: in each step of the sorting, we map our alphabet into a smaller one by denoting each sorted substring by a single letter such that the new string has no sorted substrings of length at least two (this mimics of course the molecular operation Ld).

Deciding whether a given permutation is $\mathsf{Sh} \cup \mathsf{Sd}$-sortable is of course trivial: simply test all possible sorting strategies. The problem of doing this efficiently, perhaps based on Theorems 2 and 4 remains open.

Acknowledgments. The authors were supported by the European Union project MolCoNet, IST-2001-32008. T. Harju gratefully acknowledges support by Academy of Finland, project 39802. I. Petre gratefully acknowledges support by Academy of Finland, projects 203667 and 108421, V. Rogojin gratefully acknowledges support by Academy of Finland, project 203667. G. Rozenberg gratefully acknowledges support by NSF grant 0121422.

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. 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.
4. Ehrenfeucht, A., Harju, T., Petre, I., and Rozenberg, G., Characterizing the micronuclear gene patterns in ciliates. *Theory of Comput. Syst.* **35** (2002) 501–519.
5. Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D. M., and Rozenberg, G., *Computation in Living Cells: Gene Assembly in Ciliates*, Springer (2003).
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. Harju, T., Petre, I., Li, C. and Rozenberg, G., Parallelism in gene assembly. In: *Proceedings of DNA-based computers 10*, Springer, to appear, 2005.
12. 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).
13. 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).
14. Harju, T., Petre, I., and Rozenberg, G., Modelling simple operations for gene assembly, submitted, (2005). Also as a TUCS technical report TR697, http://www.tucs.fi.
15. Jahn, C. L., and Klobutcher, L. A., Genome remodeilng in ciliated protozoa. *Ann. Rev. Microbiol.* **56** (2000), 489–520.
16. 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.
17. 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.
18. 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.
19. 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).
20. Prescott, D. M., *Cells: Principles of Molecular Structure and Function*, Jones and Barlett, Boston (1988).
21. Prescott, D. M., Cutting, splicing, reordering, and elimination of DNA sequences in hypotrichous ciliates. *BioEssays* **14** (1992) 317–324.

22. Prescott, D. M., The unusual organization and processing of genomic DNA in hypotrichous ciliates. *Trends in Genet.* **8** (1992) 439–445.
23. Prescott, D. M., The DNA of ciliated protozoa. *Microbiol. Rev.* **58**(2) (1994) 233–267.
24. Prescott, D. M., The evolutionary scrambling and developmental unscabling of germlike genes in hypotrichous ciliates. *Nucl. Acids Res.* **27** (1999), 1243 – 1250.
25. Prescott, D. M., Genome gymnastics: unique modes of DNA evolution and processing in ciliates. *Nat. Rev. Genet.* 1(3) (2000) 191–198.
26. Prescott, D. M., and DuBois, M., Internal eliminated segments (IESs) of Oxytrichidae. *J. Eukariot. Microbiol.* **43** (1996) 432–441.
27. Prescott, D. M., Ehrenfeucht, A., and Rozenberg, G., Molecular operations for DNA processing in hypotrichous ciliates. *Europ. J. Protistology* **37** (2001) 241–260.
28. Prescott, D. M., and Rozenberg, G., How ciliates manipulate their own DNA – A splendid example of natural computing. *Natural Computing* **1** (2002) 165–183.
29. Prescott, D. M., and Rozenberg, G., Encrypted genes and their reassembly in ciliates. In: M. Amos (ed.) *Cellular Computing*, Oxford University Press, Oxford (2003).