Computer Science Journal of Moldova, vol.**17**, no.3(51), 2009

On combinatorial properties of elementary intramolecular operations

Vladimir Rogojin

In loving memory of my dear father Prof. Yurii Rogozhin

Abstract

Here we tackle a problem from biology in terms of discrete mathematics. We are interested in a complex DNA manipulation process happening in eukariotic organisms of a subclass of ciliate species called *Stichotrichia* during so-called gene assembly. This process is in particular interesting since one can interpret gene assembly in ciliates as sorting of permutations. We survey here results related to studies on sorting permutations with some specific rewriting rules that formalize elementary intramolecular gene assembly operations. The research question is "what permutation may be sorted with our operations?"

Keywords: ciliates, gene assembly, elementary operations, combinatorics, molecular computing

1 Introduction

The involved process of DNA folding-recombination that takes place in ciliates during gene assembly represents high interest both for biologists as well as for computer scientists [1, 2, 3, 5, 6, 7, 9, 12, 13]. It turns out that the gene assembly process bears strong computational nature and can be thought of in such terms as linked lists, strings, permutations, graphs, etc.. Currently, the research on gene assembly in ciliates focuses both on molecular details of the process as well as on its computational aspects [16].

Ciliates are eukariotic organisms with the peculiar feature of possessing two types of nuclei called *micronucleus* and *macronucleus*. Our

focus is at a subclass of ciliates called Stichotrichia for which the features discussed in the following are especially pronounced. A macronucleus contains short gene-sized DNA molecules, where each molecule stores a single gene represented as a contiguous sequence of nucleotides. Meanwhile, a micronucleus contains long DNA organized on chromosomes, where each DNA contains many genes, each gene is broken into fragments (called MDSs), those fragments are separated by non-genetic nucleotide sequences (called IESs), MDSs are shuffled throughout the molecule, and some of the MDSs may be even inverted (for example we refer to Fig.1).

When two ciliate organisms are mating, they destroy their macronuclei and exchange the genetic material stored in their micronuclei. After the mating ciliates separate, they grow new macronuclei from some of their micronuclei. This means that ciliates assemble their macronuclear genes from the micronuclear form after the mating process occurred. During gene assembly, the micronuclear DNA molecules get transformed into the macronuclear form. During this process IESs get removed from DNA and MDSs are spliced together so that to form mature macronuclear genes (for mode details see [2]).

(a) (b)

Figure 1. From [10]. Micronuclear and macronuclear versions of *actin I* gene from *Stylonychia lemnae*. *(a) Micronuclear gene pattern:* the gene is broken into 8 MDSs. Each MDS (blue rectangle) is separated from each other by an IES (white rectangle). *MDS2* is inverted. *(b) Assembled macronuclear gene:* all IESs are removed, all MDSs are properly ordered and linked to each other to form the contiguous gene. The orientation of *MDS2* is restored.

The *intramolecular model* explains gene assembly in terms of three molecular folding-recombination operations Ld, Hi and Dlad that operate within a single molecule and splice together two or mode MDSs. Operation Ld removes an IES between two consecutive MDSs and splice the MDSs into one larger composite MDS, Hi inverts a piece of DNA

containing MDSs and IESs that results in larger composite MDS, and Dlad exchanges two pieces of DNA with places in such a way that two or more MDSs get spliced together into larger composite MDSs. Sequence of application of Ld, Hi and Dlad operations eventually transforms a micronuclear gene pattern into the macronuclear one. The folding, alignment of splicing sites and recombination of a DNA while applying Ld, Hi and Dlad is presented at Figure 2

Our focus is at the *elementary intramolecular operations*. Those are intramolecular operations that are "allowed" to invert/relocate piece of DNA containing only one micronuclear (i.e., non-composite) MDS. As the result of such a restriction, unlike the general model, elementary operations may not assemble all the micronuclear gene patterns. Moreover, for the same micronuclear gene pattern there may exist both successful (assemble the gene) as well as non-successful (fail to assemble the gene) strategies. In this way, it is not easy to answer the question what micronuclear gene patterns can be assembled into macronuclear one by elementary intramolecular operations. The goal is to find an efficient method to decide whether a micronuclear gene pattern can be assembled by elementary operations. The problem is translated into a permutation sorting problem. Here we present the recent results related to this problem.

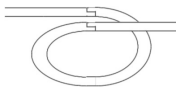
We refer to a recent review on general, simple (less restrictive than elementary) and elementary model here [8].

2 Mathematical preliminaries

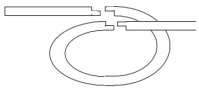
Here we recall a number of concepts and definitions from formal languages and graph theory and set the proper terminology in order to describe the following results.

By $N_{p,q}$ for some $p \leq q$ we denote set of integers $\{p, p+1, p+2, \ldots, q-1, 1\}$.

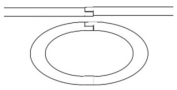
We consider a set of integers $\Pi_n = \{1, 2, \ldots, n\}$ and denote set of sequences (words) of integers from Π_n as Π_n^*. For a word v over Π_n^* we denote its domain as $dom(()v) = \{i \in \Pi_n | i \text{ occurs in } v\}$. The *signed* version of Π_n we define as $\overline{\Pi_n} = \{\bar{i} | i \in \Pi_n\}$ such that $\Pi_n \cap \overline{\Pi_n} = \emptyset$



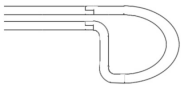
(a) Ld: the molecule folds to form a loop. A pair of recombination sites gets aligned next to each other.



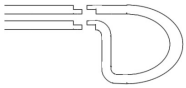
(b) Ld: enzymes enable splicing on the recombination sites.



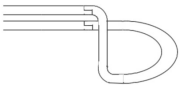
(c) Ld: the loop portion of the molecule is removed as a circular molecule.



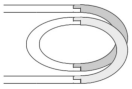
(d) Hi: the molecule folds to form a hairpin loop. A pair of recombination sites gets aligned next to each other.



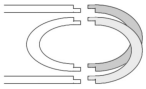
(e) Hi: enzymes enable splicing on the recombination sites.



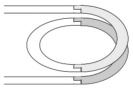
(f) Hi: the loop portion of the molecule gets inverted.



(g) Dlad: the molecule folds to form a double loop. Two pairs of recombination sites get aligned next to each other.



(h) Dlad: enzymes enable simultaneous splicing on the recombination sites.



(i) Dlad: two parts of the molecule flanked by occurrences of the recombination sites from the two different pairs exchange their places.

Figure 2. From [10]. Intramolecular recombination operations: Ld *(Loop, Direct-repeat excision)*, Hi *(Hairpin loop, Inverted-repeat excision-reinsertion)* and Dlad *(Double-Loop, Alternating Direct-repeat excision-reinsertion)*. In each case we illustrate the fold, the recombination, and the end result.

and $\bar{\cdot}$ is a bijection between sets Π_n and $\overline{\Pi_n}$. We say that $\bar{i} \in \overline{\Pi_n}$ is a signed version of $i \in \Pi_n$ and agree that $\bar{\bar{i}} = i$. For set of words over $(\Pi_n \cup \overline{\Pi_n})^*$ we consider an *inversion automorphism* as $\overline{v} = \overline{k_n} \ldots \overline{k_2 k_1}$ where $v = k_1 k_2 \ldots k_n$ is a word over $\Pi_n \cup \overline{\Pi_n}$. We also define a *sign removal operation* as a surjective mapping $||.|| : \Pi_n \cup \overline{\Pi_n} \to \Pi_n$ that for any integer $i \in \Pi_n$ let $||i|| = ||\bar{i}|| = i$. We extend this operation on words over $\Pi_n \cup \overline{\Pi_n}$ and let $||v|| = ||k_1||||k_2|| \ldots ||k_n||$.

We denote *substring* u of string v as $u \leq v$ and recall that u is called a substring of v when $v = xuy$ and $x, y, u, v \in (\Pi_n \cup \Pi_n^*)$. We denote subsequence u of string v as $u \leq_s v$ and recall that u is called a subsequence of v when $v = v_0 k_1 v_1 k_2 v_2 \ldots k_m v_m$ and $k_1, k_2, \ldots, k_n \in \Pi_n \cup \overline{\Pi_n}$, $u, v, v_0, v_1, v_2 \ldots, v_m \in (\Pi_n \cup \Pi_n^*)$. Let $s \subseteq \Pi_n \cup \overline{\Pi_n}$. Let v be a string over $\Pi_n \cup \overline{\Pi_n}$. By $v|_s$ we denote the maximal length subsequence in v that contains only letters from the subset s.

We let ϕ_k be a substring rewriting rule basing on integer $k \in \Pi_n$ and we denote its domain as $dom(\phi_k) = k$. We say that a composition of rewriting rules $\Phi = \phi_{k_m} \circ \phi_{k_{m-1}} \circ \ldots \circ \phi_{k_2} \circ \phi_{k_1}$ is a *strategy* and we denote its domain as $dom(\Phi) = \{dom(\phi_{k_1}), dom(\phi_{k_2}), \ldots, dom(\phi_{k_m})\} = \{k_1, k_2, \ldots, k_m\}$. Let v be a string over Π_n. We say that strategy Φ is applicable to v, when all operations of Φ can be applied to v in the order established by Φ such that a next operation ϕ_{k_l} from Φ applies to the result v_{l-1} produced by the preceding operation $\phi_{k_{l-1}}$, and the first operation ϕ_{k_1} applies to $v_0 = v$.

We recall that a permutation over Π_n is a bijection $\pi : \Pi_n \to \Pi_n$ that can be represented as a word $\pi = \pi(1)\pi(2)\ldots\pi(n)$. We say that a string π over $\Pi_n \cup \overline{\Pi_n}$ is a signed permutation over Π_n when $||\pi||$ is a permutation over Π_n. We regard a signed permutation π over Π_n as *sorted* if either:

- $\pi = t(t+1)\ldots n12\ldots(t-1)$, or
- $\pi = (\overline{t-1})\ldots\overline{21}\overline{n}\ldots(\overline{t+1})\bar{t}$,

where $t \in \Pi_n$. In the former case we say that π is sorted to *an orthodox order*, while in the later case we say that t is sorted to *an inverted order*. When $t = 1$ we say that π is a *linearly sorted permutation*, otherwise

we say that π is a *circularly sorted permutation*. We say that a strategy Φ that is applicable to π is a *sorting strategy* or *successful strategy* if $\Phi(\pi)$ is a sorted permutation.

We recall that a directed graph (digraph) is a structure $G = (V, E)$ that represents set of vertices V interconnected by a set of directed edges E. An alternating sequence of vertices and edges that starts and ends by two different or the same vertex is called a path in G. A path $\rho = (i_1, i_2, \ldots, i_m)$ in G starts with vertex $i_1 \in V$, ends with vertex $i_m \in V$ and contains intermediate vertices from V and edges from E that follow in orders $i_1, i_2, \ldots, i_m$ and $(i_1, i_2), (i_2, i_3), \ldots, (i_{m-1}, i_m)$ respectively. Path ρ is called a cycle when $i_1 = i_m$. Path ρ is called cyclic when $i_k = i_l$ for some $1 \leq k, l \leq m$. Also, path ρ can be regarded a a word over V.

We say that subgraph $G_R = (V_R, E_R)$ is induced by a set of paths in R in G when $V_R = \bigcup_{\rho \in R} dom(\rho)$ and $E_R = \{(i, j) | (i, j) \in \rho$ and $\rho \in R\}$.

3 Elementary gene assembly as permutation sorting

A gene pattern with n MDSs is formalized as a signed permutation π over set of integers $\Pi_n = \{1, 2, \ldots, n\}$. Here, an integer i represents the ith MDS, while signed integer $\overline{i}$ represents the inverted ith MDS in the pattern. For instance, the micronuclear gene pattern from Fig. 1 is represented as permutation $\pi = 3\,4\,5\,7\,\overline{2}\,1\,6\,8$.

Operations eh and ed formalize Hi and Dlad respectively as rewriting rules over Π_n^* [4]

- *orthodox* eh
  - $\mathsf{eh}_i(ui(\overline{i+1})v) = ui(i+1)v$;
  - $\mathsf{eh}_i(u\overline{i}(i+1)v) = ui(i+1)v$.
- *inverted* eh
  - $\mathsf{eh}_i(u(\overline{i+1})iv) = u(\overline{i+1})\overline{i}v$;

  - $\mathsf{eh}_i(u(i+1)\bar{i}v) = u(\overline{i+1})\bar{i}v$.

- *orthodox* ed
  - $\mathsf{ed}_i(uiv(i-1)(i+1)w) = uv(i-1)i(i+1)w$;
  - $\mathsf{ed}_i(u(i-1)(i+1)viw) = u(i-1)i(i+1)vw$.

- *inverted* ed
  - $\mathsf{ed}_i(u\bar{i}v(\overline{i+1})\overline{(i-1)}w) = uv(\overline{i+1})\bar{i}\overline{(i-1)}w$;
  - $\mathsf{ed}_i(u(\overline{i+1})\overline{(i-1)}v\bar{i}w) = u(\overline{i+1})\bar{i}\overline{(i-1)}vw$.

where $i \geq 1$ and $u, v, w \in \Pi_n^*$.

The process of gene assembly is represented as a permutation sorting process by eh and ed operations.

Example 1. *Let us consider signed permutation* $\pi = 1\,\overline{2}\,\overline{3}\,6\,\overline{4}\,8\,5\,10\,7\,9\,12\,11\,13$. *A sequence of* eh, ed *operations* $\Phi = \mathsf{eh}_2 \circ \mathsf{eh}_3 \circ \mathsf{ed}_6 \circ \mathsf{eh}_4 \circ \mathsf{ed}_8 \circ \circ \mathsf{ed}_{10} \circ \mathsf{ed}_{12}$ *(should be read from the right to the left) will sort* π *as follows:*

1. $\pi_1 = \mathsf{ed}_{12}(\pi) = 1\,\overline{2}\,\overline{3}\,6\,\overline{4}\,8\,5\,10\,7\,9\,11\,12\,13$;
2. $\pi_2 = \mathsf{ed}_{10}(\pi_1) = 1\,\overline{2}\,\overline{3}\,6\,\overline{4}\,8\,5\,7\,9\,10\,11\,12\,13$;
3. $\pi_3 = \mathsf{ed}_8(\pi_2) = 1\,\overline{2}\,\overline{3}\,6\,\overline{4}\,5\,7\,8\,9\,10\,11\,12\,13$;
4. $\pi_4 = \mathsf{eh}_4(\pi_3) = 1\,\overline{2}\,\overline{3}\,6\,4\,5\,7\,8\,9\,10\,11\,12\,13$;
5. $\pi_5 = \mathsf{ed}_6(\pi_4) = 1\,\overline{2}\,\overline{3}\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13$;
6. $\pi_6 = \mathsf{eh}_3(\pi_5) = 1\,\overline{2}\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13$;
7. $\pi_7 = \mathsf{eh}_2(\pi_6) = 1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13$;

The fact that Φ *transforms* π *into* π_7 *we will write as* $\Phi(\pi) = \pi_7 = 1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13$. *We note that* Φ *sorts* π *to an orthodox linearly sorted permutation.*

The following results establish the symmetric relations between orthodox and inverted eh, ed operations. First of all, if there is a strategy Φ applicable to a signed permutation π, then Φ is also applicable to $\overline{\pi}$. However, any orthodox eh, ed used while applying Φ to π changes to the inverted eh, ed when applying Φ to $\overline{\pi}$ and viceversa, any inverted eh, ed used during application of Φ to π changes to the orthodox one when applying Φ to $\overline{\pi}$ (see Theorem 1).

Theorem 1 ([4])**.** *Let π be a signed permutation over Π_n, and Φ a strategy of simple operations applicable to π. Then, Φ is applicable to $\overline{\pi}$ as well, and we have that $\overline{\Phi(\pi)} = \Phi(\overline{\pi})$.*

The following theorem (Theorem 2) says that a sortable signed permutation π cannot be sorted to an orthodox one when an inverted operation is applicable to it. Which also implies that a permutation can be sorted either to an orthodox or to an inverted form, but not to both.

Theorem 2 ([4])**.** *Permutation π is sortable to an orthodox order if and only if π is sortable and no inverted operation is applicable to π.*

Any strategy that sorts a permutation to the orthodox order will not use any of the inverted operations. Symmetrically, any strategy sorting a permutation to the inverted order will not use any of the orthodox operations. Theorem 3 establishes this fact.

Theorem 3 ([10])**.** *Let π be a signed permutation and Φ a strategy sorting π either to the orthodox or to the inverted order. If Φ sorts π to the orthodox, then Φ contains only orthodox* eh *and* ed*. Symmetrically, if Φ sorts π to the inverted, then Φ contains only inverted* eh *and* ed*.*

In this manner, it turns out that we can consider without loss of generality only orthodox eh, ed operations. Hereby, in the following sections we talk about orthodox operations only, unless specified otherwise.

Theorem 4 tells that no operation can be used more than once at a strategy applicable to a permutation π.

Theorem 4 ([4]). *Let π be a signed permutation over Π_n and $i \in \Pi_n$. Then there is no strategy applicable to π that uses either of eh_i or ed_i more than once and no strategy that would use both eh_{i-1} and ed_i.*

4 Order dependencies between eh, ed operations

The task of deciding whether a given permutation π can be sorted by eh and ed operations requires in particular that one either finds a eh, ed strategy that sorts π or at least proves its existence. As a prerequisite, one has to explore the order in which eh and ed operations can be used in a strategy applicable to π. In this section we consider so-called *dependency graphs* introduced in [10] that represent the order dependencies between orthodox eh, ed operations.

Definition 1. *[10] Let π be a signed permutation over Π_n. The* signed dependency graph *associated to π is digraph $\Gamma_\pi = (V_\pi, E_\pi)$, defined as follows: $V_\pi = dom(\pi)$, $E_\pi = E_{d-d,\pi} \cup E_{d-h,\pi} \cup E_{h-d,\pi} \cup E_{h-h,\pi}$, where*

$$
\begin{aligned}
E_{d-d,\pi} &= \{(1,1),(n,n)\} \cup \{(j,i) \mid (i-1)j(i+1) \leq_s ||\pi|| \text{ and } \\
&\quad i \leq_s \pi\} \cup \{(i,i) \mid (i+1)(i-1) \leq_s ||\pi|| \text{ or } \bar{i} \leq_s \pi\}, \\
E_{h-d,\pi} &= \{(i-2,i) \mid \overline{(i-1)} \leq_s \pi \text{ and } i \leq_s \pi\} \cup \\
&\quad \{(i+1,i) \mid \overline{(i+1)} \leq_s \pi \text{ and } i \leq_s \pi\}, \\
E_{d-h,\pi} &= \{(j,i) \mid ij(i+1) \leq_s ||\pi|| \text{ and } \\
&\quad i \text{ or } (i+1) \text{ is signed in } \pi\}, \\
E_{h-h,\pi} &= \{(i-1,i),(i+1,i) \mid \bar{i}\overline{(i+1)} \leq_s \pi \text{ for } i > 1\} \cup \\
&\quad \{(i,i) \mid (i+1)i \leq_s ||\pi|| \text{ or } i(i+1) \leq_s \pi\} \cup \\
&\quad \cup \{(2,1) \mid \text{ if } \overline{1}\,\overline{2} \leq_s \pi\} \cup \{(n,n)\}
\end{aligned}
$$

An edge $(j,i) \in E_{d-d,\pi}$ means that in any strategy applicable to π that uses orthodox ed_i the application of ed_j should precede ed_i. In the same manner, edge $(j,i) \in E_{h-d,\pi}$ means that application of orthodox ed_i is always preceded by orthodox eh_j, and edge $(i,j) \in E_{d-h,\pi}$ means that application of orthodox eh_i is always preceded by ed_j. The situation is somewhat different with edges from $E_{h-h,\pi}$ due to the fact that application of any of orthodox eh_{j-1} and of orthodox

eh_j yields j to be unsigned in π. Moreover, by the definition of the dependency graph we may have only edges of type $(i-1,i)$ and $(i+1,i)$ in $E_{h-h,\pi}$ for any $i>1$ and also egde $(2,1)$ or we may have self-loop (i,i). In other words, if $(j,i)\in E_{h-h,\pi}$ and $j\neq i$ then either $j=i-1$ or $j=i+1$ if $i>1$ and $j=2$ otherwise. Moreover, for $i>1$ and $(j,i)\in E_{h-h,\pi}$ we have that $(i-1,i),(i+1,i)\in E_{h-h,\pi}$. In this way, having edge $(j,i)\in E_{h-h,\pi}$ means that either eh_{i-1} or eh_{i+1} should precede eh_i when $i>1$ and eh_2 should precede $\mathsf{eh}_i=\mathsf{eh}_1$ otherwise. The following example demonstrates the idea.

Example 2. *Consider signed permutations* $\pi_1=1\,\overline{2}\,\overline{3}\,5\,\overline{4}\,6$ *and* $\pi_2=1\,\overline{2}\,\overline{3}\,6\,\overline{4}\,8\,5\,10\,7\,9\,12\,11\,13$. *Their dependency graphs are in Figures 3 and 4. In particular, Figure 3 tells us that in any strategy where* eh_2 *is used, either* eh_1 *or* eh_3 *should be used first. Indeed, in order to apply* eh_2*, one has to unsign either* 2 *or* 3 *in* π *first, that can be done by orthodox* eh_1 *or* eh_3 *respectively. Next, in order to use* eh_3*, one has to apply either* eh_2 *or* eh_4 *first, and also,* ed_5*. Indeed, for* eh_3 *to become applicable, one has to remove* 5 *that stays in between of* $\overline{3}$ *and* $\overline{5}$*. But that could be done by* ed_5 *only. Also, in order to use* ed_5*, one has to apply* eh_3 *first, because* ed_5 *can be applied when* 4 *is not signed, but* 4 *can be unsigned by* eh_3 *or* eh_4*. By Theorem 4 there is no strategy applicable to* π_1 *that uses both* eh_4 *and* ed_5*. Moreover, as we have established already, in order to use* eh_3 *one has to use* ed_5 *first. I.e., we have a cycle here: in order to use* ed_5 *one has to use* ed_5 *first, what by Theorem 4 is not possible. Thus,* eh_3 *and* ed_5 *cannot be used in a strategy applicable to* π*. Operation* ed_4 *cannot be used in any strategy either since one has to use either* ed_4 *or* ed_5 *before to set* 4 *and* 5 *in proper positions. In this way, subsequence of integers* 3, 4 *and* 5 *cannot be brought to the appropriate order, thus* π_1 *cannot be sorted neither to an orthodox as well as to the inverted orders. Moreover, from the graph in Figure 3 we can tell immediately that* ed_1, ed_2, ed_3, ed_4, eh_4, eh_5, ed_6 *and* eh_6 *can never be used due to the respective self-loops on the nodes.*

On the other hand, from graph in Figure 4 we see that in order to sort π_2 *we can used operations in order* ed_{12} *followed by* ed_{10}*, followed by* ed_8*, then* eh_4*, then* ed_6 *followed by* eh_3*, followed by* eh_2*.*

In this way, the dependency graphs serve well when identifying which operations should follow which operations and what operations (or sets of operations) can never be used in a strategy applicable to a signed permutation. Theorems 5, 6 and 7 illustrate these ideas.

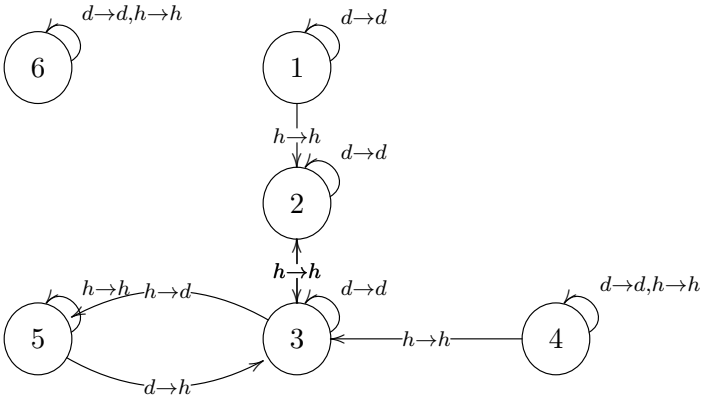


Figure 3. Dependency graphs for permutation $\pi_1 = 1\,\overline{2}\,\overline{3}\,5\,\overline{4}\,6$. The graphs include edges from: $E_{d-d,\pi}$ indicated as label $d \to d$, $E_{d-h,\pi}$ indicated as $d \to h$, $E_{h-d,\pi}$ indicated as $h \to d$, $E_{h-h,\pi}$ indicated as $h \to h$.

Theorem 5. *[10] Let π be a signed permutation over Π_n, and $\Gamma_\pi = (\Pi_n, E_{d-d,\pi} \cup E_{d-h,\pi} \cup E_{h-d,\pi} \cup E_{h-h,\pi})$ be its dependency graph. Let Φ be a strategy applicable to π, with $\Phi = \Phi_2 \circ \mathsf{ed}_i \circ \Phi_1$ and ed_i orthodox. Then, for any edge (j,i) of Γ_π, we have that:*

1. *if $(j,i) \in E_{d-d,\pi}$, then $i \neq j$ and $\Phi_1 = \Phi_1'' \circ \mathsf{ed}_j \circ \Phi_1'$;*
2. *if $(j,i) \in E_{h-d,\pi}$, then $i \neq j$ and $\Phi_1 = \Phi_1'' \circ \mathsf{eh}_j \circ \Phi_1'$, where eh_j is orthodox.*

Theorem 6. *[10] Let π be a signed permutation over Π_n, and $\Gamma_\pi = (V = \Pi_n, E_\pi = E_{d-d,\pi} \cup E_{d-h,\pi} \cup E_{h-d,\pi} \cup E_{h-h,\pi})$ be its dependency graph. Let Φ be a strategy applicable to π, with $\Phi = \Phi_2 \circ \mathsf{eh}_i \circ \Phi_1$ and eh_i orthodox. Then, for any edge $(j,i) \in E$ we have that:*

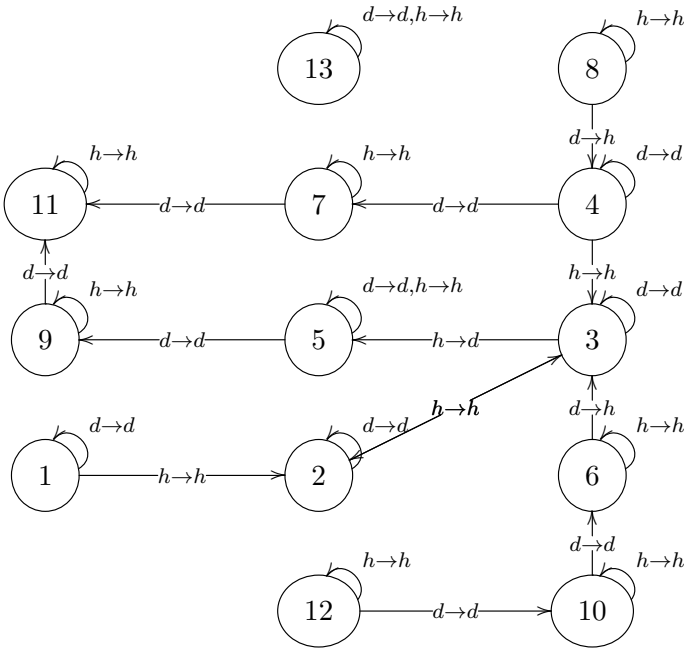


Figure 4. Dependency graphs for permutation $\pi_2 = 1\,\overline{2}\,\overline{3}\,6\,\overline{4}\,8\,5\,10\,7\,9\,12\,11\,13$. The graphs include edges from: $E_{d-d,\pi}$ indicated as label $d \to d$, $E_{d-h,\pi}$ indicated as $d \to h$, $E_{h-d,\pi}$ indicated as $h \to d$, $E_{h-h,\pi}$ indicated as $h \to h$.

- *if* $(j,i) \in E_{d-h,\pi}$, *then* $i \neq j$ *and* $\Phi_1 = \Phi_1'' \circ \mathsf{ed}_j \circ \Phi_1'$;
- *if* $(j,i) \in E_{h-h,\pi}$ *and* $i > 1$, *then* $j \in \{i-1, i+1\}$ *and* $(i-1,i),(i+1,i) \in E_{h-h,\pi}$, *and either* $\Phi_1 = \Phi_1'' \circ \mathsf{eh}_{i-1} \circ \Phi_1'$ *where* eh_{i-1} *is orthodox or* $\Phi_1 = \Phi_1'' \circ \mathsf{eh}_{i+1} \circ \Phi_1'$ *where* eh_{i+1} *is orthodox independently on* j;
- *if* $(j,i) \in E_{h-h,\pi}$ *and* $i = 1$, *then* $j = 2$ *and* $\Phi_1 = \Phi_1'' \circ \mathsf{eh}_2 \circ \Phi_1'$ *where* eh_2 *is orthodox.*

Theorem 7. *[10] Let* π *be a signed permutation,* Γ_π *its dependency graph and* $i \in dom(\pi)$.

(i) If $(i,i) \in E_{d-d,\pi}$ *then there is no strategy where orthodox* ed_i *is used in* π*;*

(ii) If $(i,i) \in E_{h-h,\pi}$ *then there is no strategy where orthodox* eh_i *is used in* π*.*

5 Deciding eh, ed-sortability

In the following simple example we show why deciding the sortability of a signed permutation by eh, ed operations may be difficult.

Example 3. *Let* $\pi = 1\,3\,5\,2\,4$ *be a signed permutation. If we apply strategy* $\Phi_1 = \mathsf{ed}_3$ *we get permutation* $\pi_1 = \Phi_1(\pi) = \mathsf{ed}_3(\pi) = 1\,5\,2\,3\,4$*. No* ed, eh *application is applicable to* π_1*, thus* Φ_1 *lead us to an unsortable permutation. However, if we apply strategy* $\Phi_2 = \mathsf{ed}_4 \circ \mathsf{ed}_2$ *we get* $\pi_2 = \Phi_2(\pi) = 1\,2\,3\,4\,5$*, what is a sorted permutation. Thus, the same permutation may have both successful and unsuccessful strategies.*

Another example demonstrates that different successful strategies can lead to different sorted permutations.

Example 4. *Let* $\pi = 2\,4\,6\,1\,3\,5$ *be a signed permutation. If we apply strategy* $\Phi_1 = \mathsf{ed}_5 \circ \mathsf{ed}_3$ *we get a circularly sorted permutation* $\pi_1 = 2\,3\,4\,5\,6\,1$*. If we use strategy* $\Phi_2 = \mathsf{ed}_4 \circ \mathsf{ed}_2$ *we get circularly sorted permutation* $\pi_2 = 6\,1\,2\,3\,4\,5$*. In this way, two different sorting strategies may lead to two different circularly sorted permutations.*

In the following we will present results concerning the decision procedure in terms of directed graphs and permutations.

In [4] we have presented a characterization of eh, ed sortable permutations in terms of directed graphs and correctly "guessed" sets.

Theorem 8. *[4] Let* π *be a unsigned permutation. Then* π *is* ed*-sortable if and only if there exists a partition* $\{1, 2, \ldots, n\} = D \cup U$*, such that the following conditions are satisfied:*

(i) $\pi|_U$ *is sorted;*

(ii) The subgraph induced by D in G_π is acyclic;

(iii) If $(p,q) \in G_\pi$ with $q \in D$, then $p \in D$;

(iv) For any $p \in D$, $(p-1)(p+1) \leq_s \pi$;

(v) For any $p \in D$, $(p-1), (p+1) \in U$.

Another theorem that characterizes eh, ed-sortable permutations requires slightly modified definition of our dependency tree that depends on some preselected sets and signed permutation rather then the signed permutation alone.

Definition 2. *[4]*
Consider a permutation π. Let $H, D \subseteq \{1, 2, \ldots, n\}$, $H \cap D = \emptyset$. The (orthodox) dependency graph $\Gamma_{\pi,H,D}$ generated by π, H and D has Π_n as its set of vertices, while its edges are defined as follows:

1. *For $q \in D$ and some $p \in \Pi_n$,*
   - *if $(q-1)\,p\,(q+1) \leq_s ||\pi||$, then $(p,q) \in \Gamma_{\pi,H,D}$;*
   - *if $(q+1)\,(q-1) \leq_s ||\pi||$, then $(q,q) \in \Gamma_{\pi,H,D}$;*
   - *if $q-1, q$ have different signs in π, then $(q-2,q) \in \Gamma_{\pi,H,D}$;*
   - *if $q, q+1$ have different signs in π, then $(q+1,q) \in \Gamma_{\pi,H,D}$.*
2. *For $q \in H$ and some $p \in \Pi_n$,*
   - *if $q\,p\,(q+1) \leq_s ||\pi||$, then $(p,q) \in \Gamma_{\pi,H,D}$;*
   - *if $(q+1)\,q \leq_s ||\pi||$ or $q\,(q+1) \leq_s \pi$, then $(q,q) \in \Gamma_{\pi,H,D}$;*
   - *if $\overline{q}(\overline{q+1}) \leq_s \pi$, then*
     - *if $q-1$ is not in H or $(q,q-1)$ is an edge, then $(q+1,q) \in \Gamma_{\pi,H,D}$,*
     - *else $(q-1,q) \in \Gamma_{\pi,H,D}$.*

For a composition Φ applicable to π, we denote $H_\Phi = \{p \in \Sigma_n \mid \mathsf{sh}_p \in \Phi\}$ and $D_\Phi = \{p \mid \mathsf{sd}_p \in \Phi\}$. Also, we denote $\Gamma_{\pi,\Phi} = \Gamma_{\pi,H_\Phi,D_\Phi}$.

The following theorem uses Definition 2 in order to characterize eh, ed sortable permutation.

Theorem 9. *[4] A permutation π is eh, ed-sortable to an orthodox order 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) For any $p \in D$, p is unsigned in π;

(ii) H sorts $\pi\,|_{H \cup U}$ to an orthodox order;

(iii) D sorts $||\pi||$;

(iv) The subgraph of $\Gamma_{\pi,H,D}$ induced by $H \cup D$ is acyclic.

The Theorems 8 and 9 require one to "guess" correctly sets D and U in order to decide the sortability for a permutation. However, in [11, 14] we have established an efficient method to decide ed sortability for a permutation without sorting it and also indicated the form of successful sorting strategies for an unsigned permutation in case it can be sorted.

The following two definitions and Theorem 10 consider integers in permutations that cannot be moved in a strategy applicable to π.

Definition 3. *[11] Let π be a signed permutation. An integer $i \in dom(\pi)$ is called forbidden if there is no strategy applicable to π that uses ed_i.*

Definition 4. *[14] Let π be a signed permutation. An integer $i \in dom(\pi)$ is called fixed if either $(i-1)i(i+1) \leq_s \pi$ or $(i+1)(i-1) \leq_s \pi$ or $\bar{i} \leq \pi$.*

Theorem 10. *[11] For a permutation π over Π_n and $p \in \Pi_n$, p is forbidden in π if and only if the subgraph $\Gamma_{\pi,p} = (T_{\pi,p}, E_{\pi,p})$ is cyclic or $q-1, q \in T_{\pi,p}$ for some q. Here $T_{\pi,p}$ is set of nodes in Γ_π from which there is a path to node p and $E_{\pi,p}$ is a subset of edges from Γ_π induced by $T_{\pi,p}$.*

We note here that set of fixed integers is also a set of forbidden integers, but not viceversa.

When knowing the set of forbidden integers that can be computed in cubic time for a signed permutation π, we can decide in linear time the sortability for π (Theorem 11).

Theorem 11. *[11]*
Permutation π is sortable if and only if $\pi|_{F(\pi)}$ is sorted.

The following definition splits Π_n into disjoint subsets called *blocks* that will help to identify successful sorting strategies.

Definition 5. *[14]*

- *A block in a signed permutation π is such set $N_{p,q}$ where p and q are fixed integers, while integers $p < i < q$ are not fixed. Set of all the blocks in π we denote as $\mathbb{B}_\pi$;*
- *A block $N_{p,q}$ is even if $q - p$ divides by 2, otherwise we say that $N_{p,q}$ is odd. Set of all even blocks in π we denote by $\mathbb{B}^e_\pi$ and set of all odd blocks we denote by $\mathbb{B}^o_\pi$;*
- *Set of all integers in even blocks we denote as B^e_π and set of all integers in odd blocks we denote as B^o_π;*
- *Set of all even integers in all blocks we denote as N^e_π and set of all odd integers in all blocks we denote as N^o_π.*

Example 5. *[14]*
Let $\pi = 1\,3\,5\,11\,7\,9\,6\,2\,4\,8\,10\,12$. Then

- $\mathbb{B}_\pi = [\{1,2,3,4,5\},\{5,6,7,8,9,10\},\{10,11,12\}]$;
- $\mathbb{B}^e_\pi = [\{1,2,3,4,5\},\{10,11,12\}]$;
- $\mathbb{B}^o_\pi = [\{5,6,7,8,9,10\}]$;
- $B^e_\pi = \{1,2,3,4,5,10,11,12\}$;
- $B^o_\pi = \{5,6,7,8,9,10\}$;
- $N^e_\pi = \{3,7,9\}$;

- $N_\pi^o = \{2, 4, 6, 8, 11\}$.

Definition 6. *[14] Let π be a permutation and let $\mathbb{B}_\pi^o = \{N_{p_1,q_1}, N_{p_2,q_2}, \ldots, N_{p_k,q_k}\}$. We define a set of subsets of $dom(\pi)$ $\mathbb{S}_\pi = \{S \in 2^{dom(\pi)} | S = (B_\pi^e \cap N_\pi^o) \cup (\bigcup_{1 \le i \le k}(N_{p_i,t_i}^o \cup N_{t_i,q_i}^e))$, where $t_i \in N_{p_i-2,q_i}^e$ for all $i, 1 \le i \le k\}$.*

Example 6. *Let us consider permutation π = 1 3 5 11 7 9 6 2 4 8 10 12 from Example 5. Here $B_\pi^e \cap N_\pi^o = \{2, 4, 11\}$, $\mathbb{B}_\pi^o = [\{5, 6, 7, 8, 9, 10\}]$. Then $\mathbb{S}_\pi = [\{2, 4, 7, 9, 11\}, \{2, 4, 6, 9, 11\}, \{2, 4, 6, 8\}]$.*

The following theorem shows that any set from $\mathbb{S}_\pi$ is the domain of a sorting strategy for permutation π, and viceversa, any strategy with the domain from $\mathbb{S}_\pi$ is a sorting strategy for π.

Theorem 12. *[14]*

Let π be a ed*-sortable permutation. Then a strategy Φ sorts π if and only if $dom(\Phi) \subseteq \mathbb{S}_\pi$ and for any $i \in dom(\Phi)$ for any $j \in T_{\pi,i} \cap dom(\Phi)$ operation ed_j is used earlier than ed_i in Φ.*

Theorems 11 and 12 state main results for deciding efficiently ed sortability for unsigned permutations. However, situation with signed permutations is more difficult because having an edge $(j, i) \in E_{h-h,\pi}$ implies choosing either eh_{i-1} or eh_{i+1} before applying eh_i. In general case this means having multiple alternatives when deciding if an operation or a set of operations can be used in a strategy applicable to π. Currently the search for an efficient decision method for eh, ed-sortability is in progress [10, 15].

Acknowledgments. V. Rogojin is on leave of absence from the Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova.

References

[1] Angela Angeleska, Nataša Jonoska, Masahico Saito, and Laura F. Landweber. RNA-guided DNA assembly. *Journal of Theoretical Biology*, 248(4):706–720, 2007.

[2] Andrzej Ehrenfeucht, Tero Harju, Ion Petre, David M. Prescott, and Grzegorz Rozenberg. *Computation in Living Cells: Gene Assembly in Ciliates*. Springer, 2004.

[3] Andrzej Ehrenfeucht, David M. Prescott, and Grzegorz Rozenberg. Computational aspects of gene (un)scrambling in ciliates. In Laura F. Landweber and Erik Winfree, editors, *Evolution as Computation*, Natural Computing Series, pages 216–256. Springer Berlin Heidelberg, 2002.

[4] Tero Harju, Ion Petre, Vladimir Rogojin, and Grzegorz Rozenberg. Patterns of simple gene assembly in ciliates. *Discrete Applied Mathematics*, 156(14):2581–2597, 2008.

[5] Lila Kari, Jarkko Kari, and Laura F. Landweber. Reversible molecular computation in ciliates. In Juhani Karhumäki, Herman Maurer, George Păun, and Grzegorz Rozenberg, editors, *Jewels are Forever*, pages 353–363. Springer Berlin Heidelberg New York, 1999.

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

[7] Laura F. Landweber and Lila Kari. Universal molecular computation in ciliates. In Laura F. Landweber and Erik Winfree, editors, *Evolution as Computation*, pages 257–274. Springer, 2003.

[8] Miika Langille, Ion Petre, and Vladimir Rogojin. Three models for gene assembly in ciliates: A comparison. *Computer Science Journal of Moldova*, 18(1):1–26, 2010.

[9] Mariusz Nowacki, Vikram Vijayan, Yi Zhou, Klaas Schotanus, Thomas G Doak, and Laura F. Landweber. RNA-mediated epigenetic programming of a genome-rearrangement pathway. *Nature*, 451(7175):153–158, 2008.

[10] Ion Petre and Vladimir Rogojin. Dependency relations between elementary intramolecular operations in gene assembly in ciliates. *Journal of Automata, Languages and Combinatorics*.

[11] Ion Petre and Vladimir Rogojin. Decision problems for shuffled genes. *Information and Computation*, 206(11):1346–1352, 2008.

[12] David M. Prescott, A. Ehrenfeucht, and G. Rozenberg. Template-guided recombination for IES elimination and unscrambling of genes in stichotrichous ciliates. *Journal of Theoretical Biology*, 222(3):323–330, 2003.

[13] David M. Prescott, Andrzej Ehrenfeucht, and Grzegorz Rozenberg. Molecular operations for DNA processing in hypotrichous ciliates. *European Journal of Protistology*, 37(3):241–260, 2001.

[14] Vladimir Rogojin. Successful elementary gene assembly strategies. *International Journal of Foundations of Computer Science*, 20(3):455–477, 2009.

[15] Vladimir Rogojin and Ion Petre. The structure of elementary strategies for gene assembly in ciliates. *Fundamenta Informaticae (submitted)*.

[16] Grzegorz Rozenberg, Thomas Bäck, and Joost N. Kok, editors. *Handbook of Natural Computing*. Springer, 2012.

Vladimir Rogojin[1]

[1]Computational Biomodelling Laboratory,
Department of Information Technologies,
Åbo Akademi University and
Turku Centre for Computer Science
Email: *vrogojin@abo.fi*