Miika Langille | Ion Petre | Vladimir Rogojin

Three models for gene assembly in ciliates: a comparison

Turku Centre *for* Computer Science

TUCS Technical Report
No 878, March 2008

Three models for gene assembly in ciliates: a comparison

Miika Langille
Department of IT, Åbo Akademi University
Turku 20520 Finland
miika.langille@abo.fi
Ion Petre
Academy of Finland and
Turku Centre for Computer Science
Department of IT, Åbo Akademi University
Turku 20520 Finland
ion.petre@abo.fi
Vladimir Rogojin
Turku Centre for Computer Science
Department of IT, Åbo Akademi University
Turku 20520 Finland
vladimir.rogojin@abo.fi

TUCS Technical Report
No 878, March 2008

Abstract

We survey in this paper the main differences among three variants of an intramolecular model for gene assembly: the general, the simple, and the elementary models. We formalize all of them in terms of sorting signed permutations and compare their behavior with respect to: (i) completeness, (ii) confluence (with the notion defined in three different setups), (iii) decidability, (iv) characterization of the sortable permutations in each model, (v) sequential complexity, and (vi) experimental validation.

TUCS Laboratory
Computational Biomodelling Laboratory

1 Introduction

Gene assembly in ciliates has been subject of intense research in the last few years, both regarding the molecular details driving it, as well as the theoretical implications of some mathematical models proposed for it, see [6, 9, 10, 16, 17, 23, 1, 20]. For a brief introduction to the biology of ciliates, especially to the gene assembly process we refer to [6]. We only recall here that ciliates have two types of nuclei: micronuclei and macronuclei. The macronuclear genes are contiguous sequences of nucleotides. The micronuclear genes on the other hand, are split into coding blocks (called MDSs), shuffled and separated by noncoding blocks (called IESs). This shuffling and inversion of MDSs is especially visible in a species of ciliates called *stichotrichs*. At some point during their life cycle, ciliates destroy all macronuclei and develop new ones from the micronuclei; in the process they must assemble correctly all coding blocks of the micronuclear genes. This process is called gene assembly.

We focus in this paper on an intramolecular model for gene assembly proposed in [9, 10] (called in the sequel the general model) and on two of its variants: the simple model, introduced in [13] and the elementary model, introduced in [15].

The general model consists of three molecular operations, ld, hi, dlad, see [9, 10], allowing the MDSs participating in an operation to be located anywhere along the molecule. Arguing on the principle of parsimony, a simplified model was introduced in [13], asking that all operations are applied 'locally'. This simple model consists of the same three molecular operations as the general model, requiring however that there is at most one coding block involved in each of the three operations. This idea was then further developed into two separate models, both using the terminology of *simple gene assembly*. In the first one, that we will refer to in here as the *elementary model*, introduced in [14, 15], the model was further restricted so that only *micronuclear*, but not *composite*, MDSs could be manipulated by the molecular operations. Consequently, once two or more micronuclear MDSs are combined into a larger composite MDS, they can no longer be moved along the sequence. The second model, that we will refer to as the *simple model* [18], allowed that both micronuclear, as well as composite MDSs may be manipulated in each of the three molecular operations.

However minor the difference between the frameworks of the simple and the elementary models may seem, it does have a great impact on the characteristics of each model. We survey in this paper the main known results on the simple and elementary gene assembly, comparing them also with the corresponding properties of the general model with respect to: (i) completeness, (ii) confluence (with the notion defined in three different setups), (iii) decidability, (iv) characterization of the sortable permutations in each model, (v) sequential complexity, and (vi) experimental validation. For this, we introduce in this paper a permutation-based presentation of the general model. We discuss in particular the question of model validation and consider the assembly of all currently known ciliate gene patterns, see [4]. We also present several open problems in this area.

2 Mathematical preliminaries

For a finite alphabet $A = \{a_1, \ldots, a_n\}$, we denote by A^* the free monoid generated by A and call any element of A^* a *word*. For any $v \in A^*$, we denote $\mathrm{dom}(v) = \{a \in A \mid a \text{ occurs in } v\}$.

Let $\overline{A} = \{\overline{a}_1, \ldots, \overline{a}_n\}$, where $A \cap \overline{A} = \emptyset$. For $p, q \in A \cup \overline{A}$, we say that p, q have the same *signature* if either $p, q \in A$, or $p, q \in \overline{A}$ and we say that they have *different signatures* otherwise. For any $u \in (A \cup \overline{A})^*$, $u = x_1 \ldots x_k$, with $x_i \in A \cup \overline{A}$, for all $1 \leq i \leq k$, we denote $\|u\| = \|x_1\| \ldots \|x_k\|$, where $\|a\| = \|\overline{a}\| = a$, for all $a \in A$. We also denote $\overline{u} = \overline{x}_k \ldots \overline{x}_1$, where $\overline{\overline{a}} = a$, for all $a \in A$. We say, that u is *uniformly signed*, if either $x_i \in A$ for all $1 \leq i \leq k$, or $x_i \in \overline{A}$ for all $1 \leq i \leq k$.

For strings u, v over Σ, we say that u is a *substring* of v, denoted by $u \leq v$, if $v = xuy$, for some strings x, y. We say that u is a *subsequence* of v, denoted by $u \leq_s v$, if $u = a_1 a_2 \ldots a_m$, $a_i \in \Sigma \cup \overline{\Sigma}$ and $v = v_0 a_1 v_1 a_2 v_2 \ldots a_m v_m$, for some strings $v_i, 0 \leq i \leq m$, over Σ.

A *permutation* π over A is a bijection $\pi : A \to A$. Fixing the order relation $(a_1, a_2, \ldots, a_m)$ over A, we often denote π as the word $\pi(a_1) \ldots \pi(a_m) \in A^*$. A *signed permutation* over A is a string $\psi \in (A \cup \overline{A})^*$, where $\|\psi\|$ is a permutation over A. We say that a signed permutation π is *(circularly) sorted* if it is of either of the following forms:

(i) $\pi = a_k a_{k+1} \ldots a_n a_1 \ldots a_{k-1}$, for some $k \geq 1$. In this case, we say that π is an *orthodox sorted permutation*.

(ii) $\pi = \overline{a_{k-1}} \ldots \overline{a_1}\, \overline{a_n} \ldots \overline{a_{k+1}}\, \overline{a_k}$, for some $k \geq 1$. In this case, we say that π is an *inverted sorted permutation*.

In both cases, if $k = 1$, then we say that π is a *linear sorted permutation*; otherwise, we say that it is *circular*.

A *sorted block* in the signed permutation π is a substring of π either of the form $a_i a_{i+1} \ldots a_j$, or of the form $\overline{a_j} \ldots \overline{a_{i+1}}\, \overline{a_i}$, $1 \leq i \leq j \leq n$, where $a_{i-1} a_i$, $\overline{a_i}\, \overline{a_{i-1}}$, $a_j a_{j+1}$, $\overline{a_{j+1}}\, \overline{a_j}$ are not substrings of π. By $S(\pi)$ we denote the total number of sorted blocks in π. Clearly, the permutation is cyclically sorted if we have $S(\pi) \leq 2$.

The notion of structure of a permutation will be useful in the paper. To define it, we first introduce the morphism $\xi_i : (A \cup \overline{A})^* \to (A \cup \overline{A})^*$, for any $1 \leq i \leq |A|$:

$$\xi_i(a_j) = \begin{cases} \lambda & \text{if } j = i; \\ a_j & \text{if } j < i; \\ a_{j-1} & \text{if } j > i; \end{cases}$$

where $a_j \in A \cup \overline{A}$.

Consider the mapping $\sigma_i : (A \cup \overline{A})^* \to (A \cup \overline{A})^*$, where for any string $u \in (A \cup \overline{A})^*$, $\sigma_i(u)$ is defined as follows:

(a) $\sigma_i(u) = u$, if $a_i a_{i+1} \not\leq u$, with $a_i, a_{i+1} \in A$, or $a_{i+1}\, a_i \not\leq u$, with $a_i, a_{i+1} \in \overline{A}$, and

(b) $\sigma_i(u) = \xi_i(u)$ otherwise.

Then, the structure of a string is the mapping $\sigma : (A \cup \overline{A})^* \rightarrow (A \cup \overline{A})^*$, such that $\sigma(u) = (\sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_{|A|-1} \circ \sigma_{|A|})(u)$. Note that the structure of a sorted permutation π is either $\sigma(\pi) = a_1$, or $\sigma(\pi) = a_2a_1$, where $a_1, a_2 \in A$, or $\sigma(\pi) = a_1a_2$, where $a_1, a_2 \in \overline{A}$.

Example 1. Consider a sorted permutation $\pi = 34512$. We find its structure $\sigma(\pi)$ as follows:

$\pi_5 = \sigma_5(\pi) = \pi$ $\quad$ $\pi_2 = \sigma_2(\pi_3) = \pi_3$

$\pi_4 = \sigma_4(\pi_5) = \xi_4(\pi_5) = 3412$ $\quad$ $\pi_1 = \sigma_1(\pi_2) = \xi_1(\pi_2) = 21$

$\pi_3 = \sigma_3(\pi_4) = \xi_3(\pi_4) = 312$ $\quad$ $\sigma(\pi) = \pi_1 = 21$

3 Gene assembly as a sorting of signed permutations

As discussed in [18, 14, 15], a natural formalization of the simple and elementary operations is through rewriting rules for signed permutations. A given gene is represented as a signed permutation by denoting the sequence and the orientation of its MDSs and assembling the gene is modeled through the sorting of the associated permutation.

For a straightforward comparison, we formalize in this paper also the general model of gene assembly [6] as a sorting of signed permutations. As observed also in the case of simple and elementary operations, it is a characteristic of permutation-based models for gene assembly that the ld operation is not explicitly modeled. Instead, it is just assumed that two consecutive blocks are going to be spliced together in a bigger composite block at some arbitrary point, independently of the other operations applied to the permutation.

3.1 Modeling of the general operations

Consider a gene pattern formalized as a signed permutation over alphabet $\Pi_n = \{1, 2, \ldots, n\}$. We formalize the general operations over signed permutations as follows:

Definition 1. i. For each $1 \leq p < n$, hi_p is defined as follows:

$$\mathsf{hi}_p(xpy(\overline{p+1})z) = xp(p+1)\overline{y}z,$$
$$\mathsf{hi}_p(x\overline{p}y(p+1)z) = x\overline{y}p(p+1)z,$$
$$\mathsf{hi}_p(x(p+1)y\overline{p}z) = x\overline{y}(\overline{p+1})\overline{p}z,$$
$$\mathsf{hi}_p(x\overline{(p+1)}ypz) = x(\overline{p+1})\overline{p}\,\overline{y}z,$$

where x, y, z are signed strings over Π_n. We denote $\mathsf{Hi} = \{\mathsf{hi}_i \mid 1 \leq i < n\}$.

ii. For each $1 \leq p, q < n$, where $|p - q| > 1$, $\mathsf{dlad}_{p,q}$ is defined as follows:

$$\mathsf{dlad}_{p,q}(xp''uq''vp'wq'z) = xwq'q''vp'p''uz,$$
$$\mathsf{dlad}_{p,q}(xp''uq'vp'wq''z) = xwvp'p''uq'q''z,$$
$$\mathsf{dlad}_{p,q}(xp'uq''vp''wq'z) = xp'p''wq'q''vuz,$$
$$\mathsf{dlad}_{p,q}(xp'uq'vp''wq''z) = xp'p''wvuq'q''z,$$

where $p' = p$, $p'' = p+1$, or $p' = (\overline{p+1})$, $p'' = \overline{p}$, and $q' = q$, $q'' = q+1$, or $q' = (\overline{q+1})$, $q'' = \overline{q}$, and x, u, v, w, z are signed strings over Π_n. In all these case, we also denote $\mathsf{dlad}_{q,p} = \mathsf{dlad}_{p,q}$.

For each $1 < p < n$, we define $\mathsf{dlad}_{p-1,p}$ and $\mathsf{dlad}_{p,p-1}$ as follows:

$$\begin{aligned}
\mathsf{dlad}_{p-1,p}(xp'''up''wp'z) &= xwp'p''p'''uz,\\
\mathsf{dlad}_{p-1,p}(xp''vp'wp'''z) &= xwvp'p''p'''z,\\
\mathsf{dlad}_{p-1,p}(xp'up'''vp''z) &= xp'p''p'''vuz,
\end{aligned}$$

where $p' = p-1$, $p'' = p$, $p''' = p+1$, or $p''' = (\overline{p+1})$, $p'' = \overline{p}$, $p' = (\overline{p-1})$, x, u, v, w, z are signed strings over Π_n. We denote $\mathsf{Dlad} = \{\mathsf{dlad}_{i,j} \mid 1 \le i, j < n, i \ne j\}$.

Example 2. Consider the permutation $\pi_1 = 2\overline{5}1\overline{4}376$. We sort it by hi and dlad as follows:

$$\begin{aligned}
&\mathsf{hi}_5(2\overline{5}1\overline{4}376) = 2\overline{7}\,\overline{3}4\overline{1}56 && \mathsf{hi}_4(\overline{4}\,\overline{7}\,\overline{3}\,\overline{2}\,\overline{1}56) = 1237456\\
&\mathsf{hi}_2(2\overline{7}\,\overline{3}4\overline{1}56) = 2374\overline{1}56 && \mathsf{dlad}_{3,6}(1237456) = 1234567\\
&\mathsf{hi}_1(2374\overline{1}56) = \overline{4}\,\overline{7}\,\overline{3}\,\overline{2}\,\overline{1}56
\end{aligned}$$

3.2 Modeling of the simple operations

Simple operations are a restriction of the general operations [7, 6]: they rearrange pieces of DNA containing at most one MDS, be that micronuclear, or composite.

Definition 2. The molecular model of simple hi and simple dlad can be formalized as follows.

i. For each $1 \le p < n$, sh_p is defined as follows:

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

where $k > i \ge 0$ and x, y are signed strings over Π_n. We denote $\mathsf{Sh} = \{\mathsf{sh}_i \mid 1 \le i \le n\}$.

ii. For each p, $2 \le p \le n-1$, sd_p is defined as follows:

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

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

Example 3. Consider the following signed permutation $\pi_1 = 5\overline{4}\,\overline{7}6\overline{3}\,\overline{1}\,\overline{2}$. It can be sorted by the following composition of simple operations

$$\begin{aligned}
&\mathsf{sh}_6(\pi) = 5\overline{4}\,\overline{7}\,\overline{6}\,\overline{3}\,\overline{1}\,\overline{2}, && \mathsf{sh}_4 \circ \mathsf{sd}_{\overline{2}} \circ \mathsf{sh}_6(\pi) = \overline{5}\,\overline{4}\,\overline{7}\,\overline{6}\,\overline{3}\,\overline{2}\,\overline{1},\\
&\mathsf{sd}_{\overline{2}} \circ \mathsf{sh}_6(\pi) = 5\overline{4}\,\overline{7}\,\overline{6}\,\overline{3}\,\overline{2}\,\overline{1}, && \mathsf{sd}_{\overline{4}} \circ \mathsf{sh}_4 \circ \mathsf{sd}_{\overline{2}} \circ \mathsf{sh}_6(\pi) = \overline{7}\,\overline{6}\,\overline{5}\,\overline{4}\,\overline{3}\,\overline{2}\,\overline{1}.
\end{aligned}$$

3.3 Modeling of the elementary operations

The elementary model is a restriction of the simple model: elementary intramolecular operations rearrange only micronuclear MDSs. This leads to the following formalization for elementary operations.

Definition 3. i. For each $p \geq 1$, eh_p is defined as follows:

$$\begin{aligned}
\mathsf{eh}_p(xp\overline{(p+1)}z) &= xp(p+1)z,\\
\mathsf{eh}_p(x\overline{p}(p+1)z) &= xp(p+1)z,\\
\mathsf{eh}_p(x(p+1)\overline{p}z) &= x\overline{(p+1)}\overline{p}z,\\
\mathsf{eh}_p(x\overline{(p+1)}pz) &= x\overline{(p+1)}\overline{p}z,
\end{aligned}$$

where x, z are signed strings over Π_n. We denote $\mathsf{Eh} = \{\mathsf{eh}_p \mid 1 \leq p \leq n\}$.

ii. For each p, $2 \leq p \leq n-1$, ed_p is defined as follows:

$$\begin{aligned}
\mathsf{ed}_p(xpy(p-1)(p+1)z) &= xy(p-1)p(p+1)z,\\
\mathsf{ed}_p(x(p-1)(p+1)ypz) &= x(p-1)p(p+1)yz,\\
\mathsf{ed}_p(x\overline{p}y\overline{(p+1)}\,\overline{(p-1)}z) &= xy\overline{(p+1)}\,\overline{p}\,\overline{(p-1)}z,\\
\mathsf{ed}_p(x\overline{(p+1)}\,\overline{(p-1)}y\overline{p}z) &= x\overline{(p+1)}\,\overline{p}\,\overline{(p-1)}yz,
\end{aligned}$$

where x, y, z are signed strings over Π_n. We denote $\mathsf{Ed} = \{\mathsf{ed}_p \mid 1 < p < n\}$.

Note that $\mathsf{Eh} \subset \mathsf{Sh} \subset \mathsf{Hi}$ and $\mathsf{Ed} \subset \mathsf{Sd} \subset \mathsf{Dlad}$.

Example 4. Consider the signed permutation $\pi = 315\overline{2}46$. It can be sorted by a composition of elementary operations as follows

$$\begin{aligned}
&\mathsf{ed}_5(\pi) = 31\overline{2}456, && \mathsf{ed}_3 \circ \mathsf{eh}_1 \circ \mathsf{ed}_5(\pi) = 123456.\\
&\mathsf{eh}_1 \circ \mathsf{ed}_5(\pi) = 312456, &&
\end{aligned}$$

3.4 Sorting strategies: terminology

A composition of operations $\Phi = \phi_k \circ \phi_{k-1} \circ \ldots \phi_2 \circ \phi_1$, where all operations are from either $\mathsf{Hi} \cup \mathsf{Dlad}$, or $\mathsf{Sh} \cup \mathsf{Sd}$, or $\mathsf{Eh} \cup \mathsf{Ed}$ is called a *strategy*. A composition $\Phi = \phi_k \circ \phi_{k-1} \circ \ldots \phi_2 \circ \phi_1$ of operations is called a *sorting strategy* for π, if $\Phi(\pi)$ is a (circularly) sorted permutation. If $\phi \in (\mathsf{Hi} \cup \mathsf{Dlad})$ for all $1 \leq i \leq k$, we say that Φ is a *general sorting strategy*. If $\phi \in (\mathsf{Sh} \cup \mathsf{Sd})$ for all $1 \leq i \leq k$, we say that Φ is a *simple sorting strategy*. If $\phi \in (\mathsf{Eh} \cup \mathsf{Ed})$ for all $1 \leq i \leq k$, we say that Φ is an *elementary sorting strategy*. We say that an unsorted signed permutation π is *blocked* if no (simple, elementary) operation is applicable to it. We say that Φ is an *unsuccessful strategy* for π, if $\Phi(\pi)$ is blocked. If there are no sorting strategies for π, then we say that π is an *unsortable permutation*.

4 Comparison of the three models

In this section we compare the general, simple and elementary intramolecular models for gene assembly by different criteria:

- completeness: whether any gene pattern may be assembled or not;

- confluence, defined in three different ways:

  (i) whether there are permutations having both successful and unsuccessful strategies,

  (ii) whether different assembly strategies starting from the same gene pattern lead to assembled genes with the same structure,

  (iii) whether different assembly strategies starting from the same gene pattern lead to the same assembled gene;

- decidability of assembly: whether it is possible to decide effectively if a given gene pattern can be assembled or not;

- characterization of gene patterns that can be assembled (starting from certain characteristics of a given gene pattern we can conclude whether the gene pattern can be assembled);

- sequential complexity is constant: whether all assembly strategies apply the same number of intramolecular operations;

- model validation: whether it is consistent with biological data.

4.1 Completeness

It was shown in [7, 6] that the general model is complete, i.e., it assembles any gene pattern. The result was proved in terms of MDS-descriptors. To prove it for signed permutations, one may take two different approaches.

On one hand, one may observe that the set of signed permutations and that of MDS descriptors are in an one-to-one correspondence. Moreover, for a signed permutation π, if $\psi(\pi)$ is its corresponding MDS descriptor, then for any operation $f \in \mathsf{Hi} \cup \mathsf{Dlad}$, $\psi(f(\pi)) = f(\psi(\pi))$. The completeness result for signed permutations then follows easily from the corresponding result for MDS descriptors.

On the second hand, one may give a direct proof of the completeness, by essentially mimicking the proof in the case of MDS descriptors. The essential observation in this case is that for any $\phi \in \mathsf{Hi} \cup \mathsf{Dlad}$ and any signed permutation π, the number of sorted blocks of $\phi(\pi)$ is smaller than that of π (i.e., $S(\phi(\pi)) < S(\pi)$). One needs to observe then that a signed permutation π is sorted if and only if $S(\pi) \leq 2$ and π is uniformly signed.

Theorem 1. *All signed permutations are sortable over* $\mathsf{Hi} \cup \mathsf{Dlad}$.

Note however that the simple and the elementary models are not complete, as shown by the following example.

Example 5. Consider the permutation $\pi = 321$. We cannot apply either eh or sh operations as all pointers have the same signature, and there is no applicable ed or sd operation either. On the other hand, π is successful in the general model: $\mathsf{dlad}_{1,2}(\pi) = 123$.

4.2 Confluence

We consider the notion of *confluence* in three different setups, so as to reflect the success of different assembly strategies, the resulting gene structure, or the resulting gene pattern. These aspects are discussed below stressing the differences between the three models for gene assembly.

Consider first the most common notion of confluence, requiring that the result of all assemblies of a given input is the same. Equivalently, all strategies for a given signed permutation are confluent. It is easy to see that neither of the three models for gene assembly is confluent in this sense. For this, consider the permutation $\pi = 2413$. Then $\mathsf{dlad}_{2,1}(\pi) = \mathsf{sd}_2(\pi) = \mathsf{ed}_2(\pi) = 4123$, while $\mathsf{dlad}_{2,3}(\pi) = \mathsf{sd}_3(\pi) = \mathsf{ed}_3(\pi) = 2341$.

The example above shows that all three models are nondeterministic in the sense that different sorting strategies may lead to different results. A natural question is then whether a given signed permutation may have both successful, as well as non-successful strategies in any of the three models. Consider then the following notion of confluence. We say that the general (simple, elementary, resp.) model is confluent if there are no signed permutations having both successful and unsuccessful strategies.

It follows from Theorem 1 that the general model is indeed confluent in the sense above. As shown in [18], the simple model is also confluent. However, the elementary model is not confluent. To see it, consider the permutation $\pi = 24135$. Then $\mathsf{ed}_3(\pi) = 23415$ is a blocked permutation, while $\mathsf{ed}_2 \circ \mathsf{ed}_4(\pi) = 12345$, a sorted permutation.

It was proved in [8, 21], see also [2], that for any gene pattern, either all general assembly strategies assemble it to a linear molecule, or all of them assemble it to a circular one. Consequently, even though if the assembly process is nondeterministic, the results of all possible assemblies of a given gene pattern have the same structure. I.e., the results of all *sorting* strategies applicable to a permutation have the same structure. As such, the same result holds also for all *sorting* strategies in the simple and in the elementary models. The question may however be asked also for the unsuccessful strategies. In this context, we say that a model for gene assembly is confluent if, for any signed permutation, all its sorting strategies lead to permutations having the same structure. Based on the considerations above, it follows easily that the general model is confluent in this sense, while the elementary model is not (since a permutation may have both successful and unsuccessful elementary strategies). Interestingly, it was proved in [18] that the simple model is in fact confluent in this sense.

Example 6. Consider permutation $\pi = 623514$. There are only two simple strategies applicable to π: $\pi_1 = \mathsf{sd}_2(\pi) = 651234$ and $\pi_2 = \mathsf{sd}_4(\pi) = 623451$. These strategies are unsuccessful, and there are no other simple strategies applicable to π. Permutation π cannot be sorted by simple operations. Note however, that permutations π_1 and π_2 have the same structure $\sigma(\pi_1) = 321 = \sigma(\pi_2)$.

The following table captures the behavior of the three models for gene assembly with respect to the three notions of confluence above. Interestingly, none of

these notions distinguishes the simple and the general model. One property that does distinguish between the two is the completeness, valid only for the general model.

	Success	Same result	Same structure
General	confluent	not confluent	confluent
Simple	confluent	not confluent	confluent
Elementary	not confluent	not confluent	not confluent

Table 1: The results of considering confluence with regard to the three aspects are summarized here.

4.3 Deciding the sortability problem

For the simple and elementary models, which are not complete, deciding the sortability of a given signed permutation is an interesting problem. Based on the confluence results in the previous section, it turns out that the problem is easy for the simple model: for any signed permutation, either all its sorting strategies are successful, or they are all unsuccessful. As such, to decide the sortability problem, it is enough to find an arbitrary strategy (e.g., using a straightforward procedure having quadratic time complexity) and answer 'yes'/'no', depending on whether or not that strategy is successful.

For the elementary model the problem of the eh-sortability of a signed permutation is easy.

Theorem 2 ([15])**.** *The unsigned permutation π is* eh*-sortable if and only if either*

(i) $\|\pi\| = k(k+1)\dots n12\dots(k-1)$ *and for some* $1 \leq i \leq k-1$, $k \leq j \leq n$ *we have* i, j *unsigned, or*

(ii) $\|\pi\| = (k-1)\dots 21n\dots(k+1)k$*, and for some* $1 \leq i \leq k-1$, $k \leq j \leq n$ *we have* i, j *signed.*

The problem of the ed-sortability turns out to be technically more involved, since a signed permutation may have both successful, and unsuccessful strategies. A complete characterization of the ed-sortable signed permutation has been given in [14, 15, 22]. The main notions used in the result are those of dependency graphs and forbidden elements. We only present here these notions for unsigned permutation; in the case of signed permutation, the setup is technically more complex, see [15]. Note also that an efficient decision procedure for the sortability problem is only known for unsigned permutation, see [22]

Dependency graphs in the elementary model

Dependency graphs suggest in which order elementary operations should be used to assemble a given gene pattern. Let π be an unsigned permutation with $\text{dom}(\pi) =$

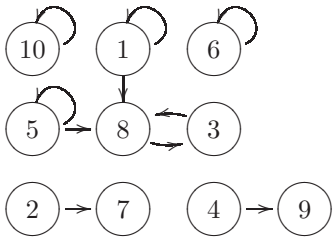

Figure 1: The dependency graph associated to $\pi = 6\,2\,8\,4\,10\,7\,1\,3\,5\,9$.

$\{1, 2, \ldots, n\}$. We associate to it a *dependency graph* $\Gamma_\pi = (V_\pi, E_\pi)$ to π, where $V_\pi = \text{dom}(\pi)$, and

$$E_\pi = \{(1,1),(n,n)\} \cup \{(i,i) | (i+1)(i-1) \leq_s \pi\} \cup \{(j,i) | (i-1)j(i+1) \leq_s \pi\}.$$

Intuitively, an edge (j, i) in Γ_π shows that in any sorting strategy for π, the operation ed_j should be used first, in order for ed_i to become applicable. If there is a loop (i, i) in Γ_π, then ed_i cannot be applied in any strategy applicable to π. We refer to [15] for a proof of these observations.

Example 7. Consider the unsigned permutation $\pi = 6\,2\,8\,4\,10\,7\,1\,3\,5\,9$. Its associated dependency graph $\Gamma_\pi = (V_\pi, E_\pi)$ is shown in Figure 1.

We have loops $(1, 1)$, $(5, 5)$, $(6, 6)$, $(10, 10)$ in the dependency graph, and so, the operations ed_1, ed_5, ed_6 and ed_{10} cannot be applied in any strategy applicable to G. We have cycle $8\,3\,8$ in Γ_π and so, neither operation ed_3, nor operation ed_8 can be applied in any strategy applicable to π. The dependency graph Γ_π suggests the following order of operations to be applied in any sorting strategy of π: ed_2 should be applied before ed_7, and ed_4 should be applied before ed_9. Indeed, for instance, strategy $\mathsf{ed}_9 \circ \mathsf{ed}_4 \circ \mathsf{ed}_7 \circ \mathsf{ed}_2(\pi)$ sorts π: $\mathsf{ed}_9 \circ \mathsf{ed}_4 \circ \mathsf{ed}_7 \circ \mathsf{ed}_2(\pi) = 6\,7\,8\,9\,10\,1\,2\,3\,4\,5$.

Forbidden elements, $\mathsf{eh}-$ and $\mathsf{ed}-$ sortability of unsigned permutations

For a signed permutation π, we say that $p \in \text{dom}(\pi)$ is *forbidden* in π if and only if there exists no composition of eh and ed operations applicable to π with p in the domain of one of them. We denote U_π the set of all forbidden elements of π. It was proved in [15] that $p \in U(\pi)$ if and only if

(i) p is on a cycle of Γ_π or

(ii) there is a path from q to p in Γ_π, for some q on a cycle of Γ_π or

(iii) there exists $r > 1$ such that there are paths from $r - 1$ to p and from r to p in Γ_π.

The following result gives the $\mathsf{eh}-$ and $\mathsf{ed}-$sortability of unsigned permutations.

Theorem 3 ([15])**.** *Let π be an unsigned permutation.*

(i) π *is* eh*-sortable if and only if either* $\|\pi\| = k(k+1)\dots n12\dots(k-1)$*, and for some* $1 \le i \le k-1$*,* $k \le j \le n$ *we have* i*,* j *unsigned, or* $\|\pi\| = (k-1)\dots 21n\dots(k+1)k$*, and for some* $1 \le i \le k-1$*,* $k \le j \le n$ *we have* i*,* j *signed.*

(ii) π *is* ed*-sortable if and only if* $\pi|_{U_\pi}$ *is sorted.*

Finding an efficient method for the eh, ed-sortability of a signed permutation remains an open problem.

4.4 Characterization of sortable permutations

The following theorem characterizes ed-sortable unsigned permutations. A similar, albeit technically more involved, characterization exists also for signed permutations, see [15].

Theorem 4 ([15])**.** *Let* π *be a unsigned permutation. Then* π *is* Ed*-sortable if and only if there exists a partition* $\{1, 2, \dots, n\} = D \cup U$*, 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 $p \in D$*,* $(p-1)(p+1) \le_s \pi$*;*

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

For simple operations we do not have a characterization of sortable permutations for the moment. For general operations the question is moot since all signed permutations are sortable.

4.5 Sequential complexity

We focus now on the length of various sorting strategies of a given signed permutation, where the length is defined as the number of operations in the strategy. Consider first the general model and let $\pi_1 = 15\overline{2}\,\overline{4}36$. One can sort it by applying $\mathsf{dlad}_{1,5} \circ \mathsf{hi}_2$, or by applying $\mathsf{hi}_2 \circ \mathsf{hi}_3 \circ \mathsf{hi}_1$. These two sorting strategies are of different length, and use a different combination of operations.

Somewhat surprisingly, the situation is different in the simple model and by consequence, also in the elementary model. It was established in [19] (using a string-based formalism) that any two sorting strategies for a given signed permutation have the same assembly length.

Theorem 5 ([19])**.** *Let* π *be a signed permutation and* ϕ, ψ *be two simple sorting strategies for* π*. Then* ϕ *and* ψ *have the same sequential assembly length. Moreover, they have the same number of* sh *and the same number of* sd *operations.*

The differences between the general model and the two restricted models go beyond Theorem 5. E.g., when choosing operations in the simple model, we may always just choose the first available operation as the number of operations required in the end remains the same. If the operations were given different weights or costs, then the general model may have optimal and sub-optimal sorting strategies. We refer to [12] for a detailed discussion on various measures of complexity for gene assembly.

4.6 Model validation

A database of known sequences of micronuclear and macronuclear ciliate genes can be found in [4]. Based on the completeness result for the general model, it is clear that all the gene patterns have an assembly strategy in the general model. As it turns out however, the elementary model cannot account for the assembly of some of the gene patterns in [4].

Example 8. Actin I gene in it Sterkiella nova is represented by the permutation $\pi = 346579\overline{2}18$. It is easy to check that there is no elementary sorting strategy applicable to π. However, we can sort π by applying the simple sorting strategy

$$\mathsf{sh}_1 \circ \mathsf{sh}_2 \circ \mathsf{sd}_8 \circ \mathsf{sd}_5(\pi) = \overline{9}\,\overline{8}\,\overline{7}\,\overline{6}\,\overline{5}\,\overline{4}\,\overline{3}\,\overline{2}\,\overline{1}.$$

Below we will outline all the available scrambled gene patterns in [4], together with one simple sorting strategy. Genes that are not scrambled in their micronuclear form or the ones that have missing MDSs will not be included.

Actin I, Sterkiella nova : $\pi = 346579\overline{2}18$;

$$\mathsf{sh}_1 \circ \mathsf{sh}_2 \circ \mathsf{sd}_8 \circ \mathsf{sd}_6(\pi) = \overline{987654321}.$$

Actin I, Sterkiella histriomuscorum : $\pi = 346579\,10\,\overline{2}18$;

$$\mathsf{sh}_1 \circ \mathsf{sh}_2 \circ \mathsf{sd}_8 \circ \mathsf{sd}_6(\pi) = \overline{10}\,\overline{987654321}.$$

Actin I, Stylonychia pustulata : $\pi = 346578\overline{2}1$;

$$\mathsf{sh}_1 \circ \mathsf{sh}_2 \circ \mathsf{sd}_6(\pi) = \overline{87654321}.$$

α Telomere Binding Protein, Sterkiella nova :

$$\pi = 1\,3\,5\,7\,9\,11\,2\,4\,6\,8\,10\,12\,13\,14;$$
$$\mathsf{sd}_{10} \circ \mathsf{sd}_8 \circ \mathsf{sd}_6 \circ \mathsf{sd}_4 \circ \mathsf{sd}_2(\pi) = 1\,2\,3\,4\,5\,6\,7\,8\,9\,10\,11\,12\,13\,14.$$

DNA Polymerase α, Paraurostyla weissei:

$\pi = \overline{44}\,\overline{42}\,\overline{40}\,\overline{38}\,\overline{36}\,\overline{34}\,\overline{32}\,\overline{30}\,\overline{28}\,\overline{26}\,\overline{24}\,\overline{22}\,\overline{20}\,\overline{18}\,\overline{16}\,\overline{14}\,\overline{12}\,\overline{10}\,\overline{8}\,\overline{6}\,1\,2\,3\,4\,5\,7\,9\,11\,13$
$15\,17\,19\,21\,23\,25\,27\,29\,31\,33\,35\,37\,39\,41\,43\,45\,46\,47\,48$

The signed permutation sorting strategy for this gene is just sh_1 repeated 40 times.

4.7 Summary

The following table summarize properties of general, simple and elementary models considered in this paper.

	General	Simple	Elementary
Completeness	complete	not complete	not complete
Confluence (Success)	confluent	confluent	not confluent
Confluence (Structure)	confluent	confluent	not confluent
Confluence (Result)	not confluent	not confluent	not confluent
Deciding Sortability	trivial	confluence (success)	forbidden elements
Characterizing sortable permutations	not a problem	open problem	dependency graph
Sequential Complexity is Constant	no	yes	yes
Model Validation	unknown	valid	not valid

Table 2: Summary for general, simple and elementary intramolecular models

5 Open problems

There are two currently open problems related to the simple model: the linear decidability of the sortability problem and computing the number of sortable permutations of length n. It is however possible that these two problems are intertwined and an answer to one may at least partly solve the other.

Decidability. It was shown in [18] that it is possible to decide whether a permutation is sortable or unsortable in the simple model by applying available operations in an arbitrary order until the permutation is blocked or sorted. This gives us a quadratic method for deciding. Our first open problem is related to the optimality of this method: is there a procedure to decide in linear time the sortability problem in the simple model?

For the elementary model, finding an efficient decision procedure for $\{\mathsf{eh}, \mathsf{ed}\}$-sortability problem is also open.

Sortable permutations of length n. As we pointed out also in this paper, not all permutations may be sorted using the simple operations. This differs from the general model which has been shown to be complete. Thus, an interesting problem is computing how many permutations of length n are sortable in the simple/elementary models. As a related problem, it should even be interesting to see whether the ratio of sortable signed permutations tends to 0 when n tends to infinity. Both problems are open also in the case of unsigned permutations.

Acknowledgments. Miika Langille, Ion Petre and Vladimir Rogojin are supported by Academy of Finland, project 108421. Vladimir Rogojin is on leave of absence from Institute of Mathematics and Computer Science of Academy of Sciences of Moldova, Chisinau MD-2028 Moldova. Vladimir Rogojin is supported by Science and Technology Center in Ukraine, project 4032.

References

[1] Angeleska, A., Jonoska, N., Saito, M., and Landweber, L.F., RNA-Template Guided DNA Assembly. *Journal of Theoretical Biology* **248**, Elsevier (2007), 706–720.

[2] Brijder, R., Hoogeboom, H.J., Rozenberg, G., Reducibility of gene patterns in cliates using the breakpoint graph, *Theoret. Comput. Sci* **356** (2006) 26–45

[3] Brijder, R., Langille, M. and Petre, I., A String-based Model for Simple Gene Assembly. In: E. Csuhaj-Varju, Z. Esik (Eds.) *Proceedings of FCT 2007*, Lecture Notes in Computer Science, **4639**, Springer-Verlag Berlin Heidleberg, (2007), pp. 161-172.

[4] 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), pp. 396–398.

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

[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. Circularity and other invariants of gene assembly in ciliates. 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] Prescott, D. M., Ehrenfeucht, A., and Rozenberg, G., Molecular operations for DNA processing in hypotrichous ciliates. *Europ. J. Protistology* **37** (2001), pp. 241–260.

[11] Harju, T., Li, C., Petre, I. and Rozenberg, G., Parallelism in gene assembly, *Natural Computing*, 5 (2) (2006), pp. 203-223.

[12] Harju, T., Li, C., Petre, I. and Rozenberg, G., Complexity Measures for Gene Assembly. In: K. Tuyls (ed.) *Proceedings of the Knowledge Discovery and Emergent Complexity in Bioninformatics workshop*, Springer, Lecture Notes in Bioinformatics 4366, (2007), pp 42-60.

[13] Harju, T., Petre, I., and Rozenberg, G., Modelling simple operations for gene assembly. In: J.Chen, N.Jonoska, G.Rozenberg (Eds.) *Nanotechnology: Science and Computation* (2006), pp. 361–376.

[14] Harju, T., Petre, I., Rogojin, V. and Rozenberg, G., Simple operations for gene assembly. In: L. Kari (eds.) *Proceedings of DNA-based computers 11*, Lecture Notes in Computer Science, **3892**, Springer, (2006), pp. 96–111.

[15] Harju, T., Petre, I., Rogojin, V. and Rozenberg, G., Patterns of Simple Gene Assembly in Ciliates, *Discrete Applied Mathematics*, Elsevier, to appear (2007).

[16] 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.

[17] 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).

[18] Langille, M., Petre, I., Simple gene assembly is deterministic. *Fundamenta Informaticae* **72**, IOS Press, (2006), pp. 1–12.

[19] Langille, M., Petre, I., Sequential vs. Parallel Complexity in Simple Gene Assembly. *Theoretical Computer Science*, Elsevier B.V., (2007), to appear.

[20] Nowacki, M., Vijayan, V., Zhou, Y., Schotanus, K., Doak, T.G., Landweber, L.F., RNA-mediated epigenetic programming of a genome-rearrangement pathway. *Nature* **451**, doi:10.1038/nature06452, (2008) 153–158.

[21] Petre, I., Invariants of gene assembly in stichotrichous ciliates. *IT*, Oldenbourg Wissenschftsverlag, **3** (2006), pp. 161–167.

[22] Petre, I., and Rogojin, V., Decision Problem for Shuffled Genes, *Information and Computation*, (2007), submitted.

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

Lemminkäisenkatu 14 A, 20520 Turku, Finland | www.tucs.fi

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 978-952-12-2056-2
ISSN 1239-1891