Computational nature of gene assembly in ciliates

Robert Brijder[1], Mark Daley[2], Tero Harju[3], Natasha Jonoska[4], Ion Petre[5], and Grzegorz Rozenberg[1,6]

[1] Leiden Institute of Advanced Computer Science, Universiteit Leiden, Niels Bohrweg 1, 2333 CA Leiden, the Netherlands, rbrijder,rozenber@liacs.nl
[2] Department of Computer Science, University of Western Ontario, London, Ontario, Canada, daley@csd.uwo.ca
[3] Department of Mathematics, University of Turku, Turku 20014, Finland, harju@utu.fi
[4] Department of Mathematics, University of South Florida, jonoska@math.usf.edu
[5] Åbo Akademi University, Turku 20520 Finland ion.petre@abo.fi
[6] Department of Computer Science, University of Colorado at Boulder, USA

1 Introduction

The mathematical study of gene assembly in ciliates was initiated in [58] and in [59], where it was noted that the DNA rearrangements performed by ciliates have a strong computational appeal. The first results dealt with the computational capabilities of suitably defined models for gene assembly, using classical approaches from theoretical computer science, especially based on formal languages and computability theory. Shortly afterwards, a parallel line of research was initiated in [33] and in [75], where the focus was to study various properties of the gene assembly process itself, understood as an information processing process that transforms one genetic structure into another.

This research area has witnessed an explosive development, with a large number of results and approaches currently available. Some of them belong to computer science: models based on rewriting systems, permutations, strings, graphs, and formal languages, invariants results, computability results, etc., see, e.g., [29], while others, such as template-based DNA recombination, belong to theoretical and experimental biology, see, e.g., [34] and [5].

In this chapter we review several approaches and results in the computational study of gene assembly. In Section 2 we introduce the basic biological details of the gene assembly process as currently understood and experimentally observed. After mathematical preliminaries in Section 3, we introduce two of the mostly studied molecular models for gene assembly (intermolecular and intramolecular) in Sections 4 and 5. We also discuss several mathematical approaches used in studying these models. In Section 6 we discuss some properties of the gene assembly process, called invariants, that hold independently of molecular model and assembly strategy. In Section 7 we present models for template-based DNA recombination as a possible

molecular implementation of the gene assembly process. We conclude the chapter with a brief discussion in Section 8.

2 The basic biology of gene arrangement in ciliated protozoa

All living cells can be classified as belonging to one of two high-level groups: the *prokaryotes* or the *eukaryotes*. The prokaryotes are defined chiefly by their simple cellular organization: within a prokaryotic cell there are no compartments or subdivisions, all of the intracellular materials (e.g., enzymes, DNA, food, waste) are contained within a single cellular membrane and are free to intermix. By contrast, eukaryotic cells contain nested membranes with many functionally and morphologically distinct organelles, the most well-known being the nucleus which contains the DNA, and DNA processing machinery. All bacteria and archaea are prokaryotes while all higher multicellular organisms are eukaryotes.

The *protozoa* are single-celled eukaryotes of striking complexity; each cell functions as a complete, individual organism capable of advanced behaviors typically associated with multi-cellular organisms. Protozoa are equipped with extensive sensory capabilities and various species can sense temperature, light and motion as well as chemical and magnetic gradients. In addition to locomotion via swimming, some types of ciliated protozoa are able to coordinate legs made from fused cilia to walk along substrates in search of food. Many protozoan species are active hunters and possess elementary decision-making abilities enabling them to identify, hunt and consume prey.

Protozoa are found in nearly every habitat on Earth, and form a critical portion of the microbial food web. Beyond such ecological significance, the study of protozoa is fundamental to evolutionary inquiry as the protozoa represent a significant portion of eukaryotic evolutionary diversity on Earth – through their evolution they have produced unique features, one of which we study further in this chapter.

The *ciliated protozoa* (phylum Ciliophora) are a particularly interesting group due to their unique, and complex, nuclear morphology and genetics. Where most eukaryotic cells contain only one type of nucleus (sometimes in many copies), ciliate cells contain two functionally different types of nucleus: *Macronucleus* (abbreviated as *MAC*) and *Micronucleus* (*MIC*), each of which may be present in various multiplicities. For details on the many other aspects of ciliate biology which we are not able to touch upon here, we refer to [47, 73].

The diversity within the phylum Ciliophora is staggering and even relatively closely related species have extreme evolutionary distances, e.g., the ciliates *Tetrahymena thermophila* and *Paramecium caudatum* have an evolutionary distance roughly equivalent to that between rat and corn [71]. In this chapter we restrict our discussion to ciliates of the subclass Stichotrichia for which the unique features that we discuss in this chapter are especially pronounced.

In Stichotrichia there is a profound difference in genome organization between the MIC and MAC on both a global level (where one considers the organization

of chromosomes) and a local level (considering the organization of DNA sequence within individual genes).

On the global level, the MIC is diploid (there are exactly two copies of each chromosome) and composed of long chromosomes containing millions of base pairs of DNA. Each of these chromosomes contains a large number of genes, although these genes account for only a very small portion of the total sequence of the chromosomes (approximately 2-5% in stichotrichs). This long, but genetically sparse, chromosome structure is similar to that found in the nuclei of most other eukaryotes.

The global organization of the MAC provides a stark contrast to that of the MIC: most obviously, the number of copies of chromosomes in MACs is very large. For example, typically more than 1000 copies in stichotrichs, more than 10000 in *Stylonichia*, and at the extreme end up to millions of copies of the rDNA-containing chromosome in the stichotrich *Oxytricha trifallax* (also called *Sterkiella histriomuscorum*). Along with much higher number of copies, MAC chromosomes are much shorter than those of the MIC and they contain only 1–3 genes each. Although they are small, these chromosomes are genetically dense – indeed, approximately 85% of a typical chromosome sequence contain genes. Hence, although we have a huge multiplicity of these chromosomes, the total DNA content is still much lower than in the MIC. Thus, the MIC chromosomes are long and "sparse" while the MAC chromosomes are short and "dense".

The difference in global genetic organization between the MAC and MIC is impressive, but turns out to be much less startling than the difference in the local organization of the genes. On the local level, the MAC is a functional nucleus and, like most eukaryotic nuclei, it carries out the day-to-day genetic "housekeeping" tasks of the cell including the production of proteins from the functional MAC genes. In contrast, the genes of the MIC are not functional (not expressible as proteins) due to the presence of many non-coding sequences which break up the genes. Indeed, the macronuclear forms of ciliate genes are heavily modified from the original micronuclear configurations. MIC genes can be divided into two interleaving types of regions: *Macronuclear Destined Segments* or *MDSs* and *Internal Eliminated Sequences* or *IESs*. The MDSs are the regions of the MIC gene that end up in the functional MAC version of the gene, while the IESs are interspersed non-genetic regions of sequence that do not appear in the MAC version of the gene, see Fig. 1. The MDSs are assembled, via overlapping regions called *pointers* to form the macronuclear genes. Each MDS, with the exception of the first and last, is flanked on either side by one of these pointer regions. The outgoing pointer region on one side, which we depict at the right side of MDS n has identical DNA sequence to the incoming pointer region on the left side of MDS $n+1$. An illustration of this can be seen in Fig. 2; note that the outgoing pointer of MDS n is identical to the incoming pointer of MDS $(n+1)$.

In some ciliates, in addition to the appearance of IESs, the MIC genes have the further complication that the MDSs do not appear in the same order as they do in the functional MAC gene. That is, the order of MDSs in the MIC gene is *scrambled* relative to their "orthodox" order in the MAC gene and may even contain segments which are inverted (i.e., rotated 180 degrees). A schematic representation of the gene

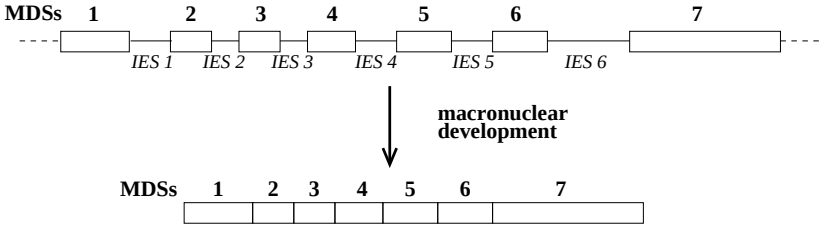


Fig. 1. A diagram of the arrangement of six IESs and seven MDSs in the micronuclear (top) and the macronuclear (bottom) gene encoding βTP protein. During macronuclear development the IESs are excised and the MDSs are ligated (by overlapping of the ends) to yield a macronuclear gene. MDSs are *rectangles*. IESs are *line segments* between *rectangles*. [74]

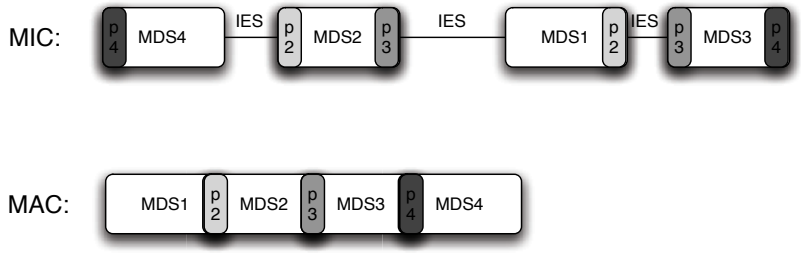


Fig. 2. Schematic representation of MIC gene (top) and associated MAC gene (bottom) with pointers indicated on the MDSs.

actin I from *O. trifallax* is shown in Fig. 3 (see [77]). Note that the numbers enumerating MDSs refer to the orthodox order of the MDSs in the macronuclear version of the gene, and a bar is used to denote MDSs which are inverted.

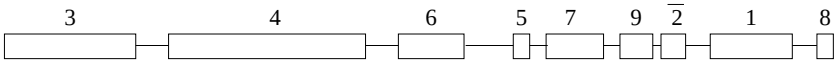


Fig. 3. Schematic representation of the structure of the micronuclear gene-encoding actin protein in the stichotrich *Sterkiella nova*. The nine MDSs are in a scrambled disorder. [77]

Ciliates reproduce asexually but, during times of environmental stress (e.g., starving due to lack of nutrients) can also undergo the (non-reproductive) sexual activity of *conjugation*. Rather than producing offspring, the purpose of conjugation in ciliates is to give each cell a "genetic facelift" by incorporating new DNA from the conjugating partner cell. The specifics of conjugation are highly species-dependent, but almost all ciliate species follow the same basic outline, the generic form of which is illustrated in Fig. 4. Two cells form a cytoplasmic bridge while meiotically dividing

their diploid MICs into four haploid MICs. Each cell sends one haploid MIC across the bridge to the conjugating partner. The cell then combines one of its own haploid MICs with the newly-arrived haploid MIC from the partner to form a new fused diploid MIC. Also, each cell destroys its old MAC and remaining haploid MICs, the fused MIC divides into two parts, one of which will be the new MIC and the other one will develop into (will be transformed into) the new MAC.

The topic of interest in this chapter is this transformation of a new MIC into a new MAC. We focus in particular on the transformations of the individual genes – a process called *gene assembly*. This process is fascinating both from the biological and information processing points of view. Indeed, gene assembly is the most complex example of DNA processing known to us in nature, underscored by the dramatic difference in the structure, and composition, of the MIC and MAC genomes explained above. To form a functional macronuclear gene, all IESs must be excised, all MDSs must be put in their original (orthodox) order with inverted segments switched to their proper orientation.

Understanding and investigating the computational nature of the gene assembly process, and its biological implications, becomes the central focus of this chapter. For further details on the biology of gene assembly we refer to [53][72][73][65] [34] and, in particular to [29], which contains chapters explaining basic biology, basic cell biology and the basic biology of ciliates written specifically for motivated computer scientists.

3 Mathematical preliminaries

In this section we fix basic mathematical notions and terminology used in this chapter.

3.1 Strings

For an alphabet Σ, let Σ^* denote the set of all strings over Σ. Let Λ denote the *empty string*. For a string $u \in \Sigma^*$, we denote by $|u|$ its length, i.e., the number of letters u consists of.

A string u is a *substring* of a string v, if $v = xuy$, for some $x, y \in \Sigma^*$. In this case, we denote $u \leq v$. We say that u is a *conjugate* of v if $v = w_1 w_2$ and $u = w_2 w_1$, for some $w_1, w_2 \in \Sigma^*$.

For an alphabet Σ, let $\overline{\Sigma} = \{\overline{a} \mid a \in \Sigma\}$ be a signed disjoint copy of Σ. The set of all strings over $\Sigma \cup \overline{\Sigma}$ is denoted by $\Sigma^{\maltese} = (\Sigma \cup \overline{\Sigma})^*$. A string $v \in \Sigma^{\maltese}$ is called a *signed string over* Σ. We adopt the convention that $\overline{\overline{a}} = a$ for each letter $a \in \Sigma$.

Let $v \in \Sigma^{\maltese}$ be a signed string over Σ. We say that a letter $a \in \Sigma \cup \overline{\Sigma}$ *occurs* in v, if a or $\overline{a}$ is a substring of v. Let $\mathsf{dom}(v) \subseteq \Sigma$, called the *domain* of v, be the set of the (unsigned) letters that occur in v.

Example 3.1. For the alphabet $\Sigma = \{2, 3\}$ of pointers, $\overline{\Sigma} = \{\overline{2}, \overline{3}\}$. Here $u = 2\,3\,3 \in \Sigma^* \subseteq \Sigma^{\maltese}$, while $v = 2\,\overline{3}\,\overline{2} \in \Sigma^{\maltese}$ is a signed string over Σ for which $\mathsf{dom}(v) = \{2, 3\}$ although 3 is not a substring of v.

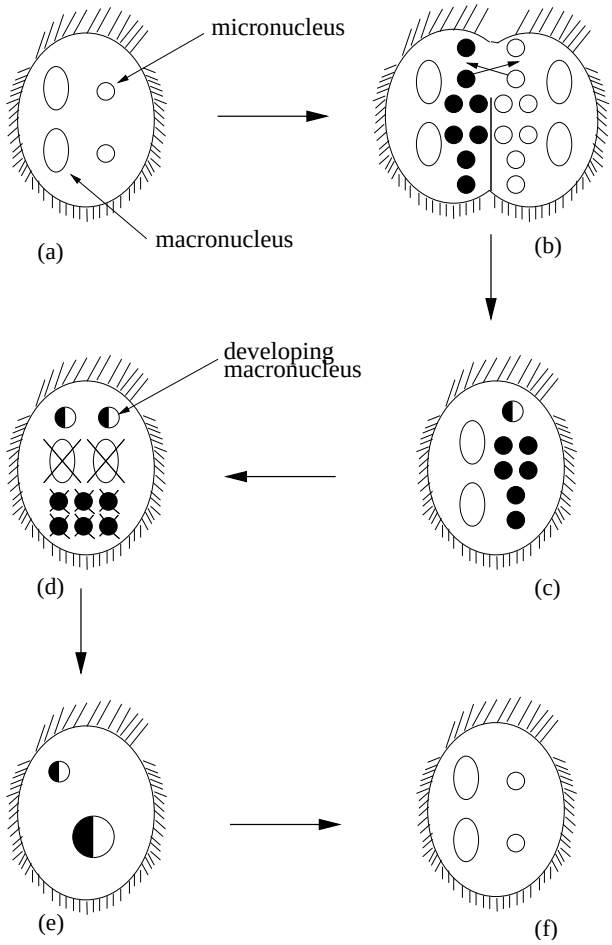


Fig. 4. a–f. Conjugation in stichotrichs: **a** A stichotrich with two macronuclei and two micronuclei. **b** Two stichotrichs have joined and formed a cytoplasmic channel. The two diploid micronuclei have each formed four haploid micronuclei. **c** The two cells exchanged haploid micronuclei, and the two organisms have separated. The exchanged haploid micronucleus has fused with a resident haploid micronucleus forming a new diploid micronucleus (half white and half black). **d** The new diploid micronucleus has divided by mitosis. The unused haploid micronuclei (six) and the two macronuclei are degenerating. **e** One of the new daughter micronuclei has developed into a new macronucleus. The old macronucleus and the unused haploid micronuclei have disappeared. **f** Conjugation has been completed. The micronucleus and macronucleus have divided, yielding the appropriate nuclear numbers (in this case, two MICs and two MACs). [29]

The signing $a \mapsto \overline{a}$ of letters extends to strings in a natural way: for a signed string $u = a_1 a_2 \dots a_n \in \Sigma^{\maltese}$, with $a_i \in \Sigma \cup \overline{\Sigma}$ for each i, let the *inversion* of u be

$$\overline{u} = \overline{a}_n \overline{a}_{n-1} \dots \overline{a}_1 \in \Sigma^{\maltese} .$$

For any set of strings $S \subseteq \Sigma^{\maltese}$, we denote $\overline{S} = \{\overline{u} \mid u \in S\}$. For two strings $u, v \in \Sigma^{\maltese}$, we say that u and v are *equivalent*, denoted $u \approx v$, if u is a conjugate of either v or $\overline{v}$.

For an alphabet Σ, let $\|.\|$ be the substitution that unsigns the letters: $\|a\| = a = \|\overline{a}\|$. Mapping $\|.\|$ extends to $\Sigma^{\maltese}$ in the natural way. A signed string v over Σ is a *signing* of a string $u \in \Sigma^*$, if $\|v\| = u$. A signed string u, where each letter from Σ occurs exactly once in u, is called a *signed permutation*.

For two alphabets Σ and Δ, a mapping $f : \Sigma^{\maltese} \to \Delta^{\maltese}$ is called a *morphism* if $f(uv) = f(u)f(v)$ and $f(\overline{u}) = \overline{f(u)}$. If $\Delta \subseteq \Sigma$, a morphism $f : \Sigma^* \to \Delta^*$ is called a *projection* if $f(x) = x$ for $x \in \Delta$ and $f(x) = \lambda$, for $x \in \Sigma \setminus \Delta$.

Example 3.2. For the alphabet $\Sigma = \{2, 3, 4, 5\}$, there are $2^4 \cdot 4! = 384$ signed permutations. The signed strings $2\,3\,4\,5$ and $\overline{4}\,\overline{2}\,5\,3$ are among them.

Throughout this chapter we call $\{2, 3, \dots\}$ the set of *pointers*.

3.2 Graphs

For a finite set V, let $E(V) = \{\{x, y\} \mid x, y \in V,\ x \neq y\}$ be the set of all unordered pairs of different elements of V. A (*simple*) *graph* is an ordered pair $G = (V, E)$, where V and E are finite sets of *vertices*, and *edges*, respectively. If $e = \{x, y\} \in E$ is an edge, then the vertices x and y are the *ends* of e. In this case, x and y are *adjacent* in G. If $V = \varnothing$, then we denote $G = \varnothing$ and call it the *empty graph*.

For a vertex x in G, let $N_G(x) = \{y \mid \{x, y\} \in E\}$ be the *neighborhood* of x in G. A vertex x is *isolated* in G, if $N_G(x) = \varnothing$.

A *signed graph* $G = (V, E, \sigma)$ consists of a simple graph (V, E) together with a labeling $\sigma\colon V \to \{-, +\}$ of the vertices. A vertex x is said to be *positive*, if $\sigma(x) = +$; otherwise x is *negative*. We write $x^{\sigma(x)}$ to indicate that the vertex x has sign $\sigma(x)$.

Let $G = (V, E, \sigma)$ be a signed graph. For a subset $A \subseteq V$, its *induced subgraph* is the signed graph $(A, E \cap E(A), \sigma)$, where σ is restricted to A. The *complement* of G is the signed graph $G^c = (V, E(V) \setminus E, \sigma^c)$, where $\sigma^c(x) = +$ if and only if $\sigma(x) = -$. Also, let $\mathsf{loc}_x(G)$ be the signed graph obtained from G by replacing the subgraph induced by $N_G(x)$ by its complement. For a subset $A \subseteq V$, we denote by $G - A$ the subgraph induced by $V \setminus A$.

A *multigraph* is a (undirected) graph $G = (V, E, \varepsilon)$, where parallel edges are possible. Therefore, E is a finite set of edges and $\varepsilon : E \to \{\{x, y\} \mid x, y \in V\}$ is the *endpoint mapping*. We allow $x = y$, and therefore edges can be of the form $\{x, x\} = \{x\}$ – an edge of this form should be seen as a 'loop' for x.

4 The intermolecular model for gene assembly

The very first formal model of the ciliate gene assembly process was the *intermolecular model* proposed in [58, 59]. We introduce in this section a string-based formalization of the model and refer to [58] for the biological details of the model. Given a string $uxvxw$, where u, x, v, w are nonempty substrings, the authors defined the following intramolecular operation: $uxvxw \rightarrow \{uxw, \cdot vx\}$ where $\cdot$ denotes that the string vx is circular. Intuitively, this models the action of a strand of DNA, $uxvxw$, looping over onto itself and aligning the regions containing the subsequence x. With the x's aligned, the DNA strand can undergo recombination, yielding the two products uxw and $\cdot vx$. Likewise, the inverse, intermolecular, operation was also defined: $\{uxw, \cdot vx\} \rightarrow uxvxw$. These operations are illustrated in Fig. 5.

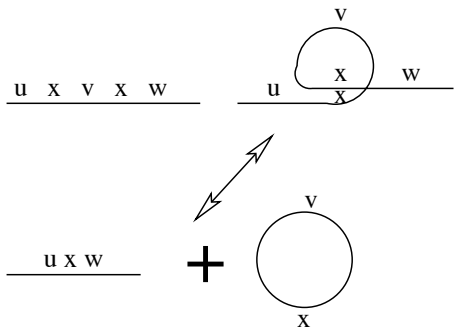


Fig. 5. The (reversible) recombination of $uxvxw \leftrightarrow \{uxw, \cdot vx\}$ in the intermolecular model for gene assembly of [58, 59].

Example 4.1. Consider a hypothetical micronuclear gene with 4 MDSs where the MDSs come in the order $M_1M_2M_4M_3$. If we denote each MDS by its pair of incoming/outgoing pointers and/or markers, we can then denote the whole micronuclear gene as $\delta = (b, p_2)(p_2, p_3)(p_4, e)(p_3, p_4)$. (For more details on formal representation of ciliate genes, see Section 5.) An assembly strategy for this gene in the intermolecular model is the following (we indicate for each operation the pointer on which the molecule is aligned):

$$\begin{aligned}\delta &\xrightarrow{p_3} (b, p_2)(p_2, p_3, p_4) \ + \ \cdot(p_4, e)p_3 \xrightarrow{p_4} (b, p_2)(p_2, p_3, p_4, e)p_3p_4 \\ &\xrightarrow{p_2} (b, p_2, p_3, p_4, e)p_3, p_4 \ + \ \cdot p_2.\end{aligned}$$

As indicated by the formal notation above, the gene gets assembled with copies of pointers p_3 and p_4 following the assembled gene and a copy of pointer p_2 placed on a separate circular molecule. Note that the notation above ignores all IESs of the micronuclear and of the assembled gene.

The obvious difficulty with the intermolecular model is that it cannot deal with DNA molecules in which a pointer is inverted – this is the case, e.g., for the actin I gene in *S. nova*. Nevertheless, we can show that inverted pointers can be handled in this model, provided the input molecule (or its MDS-IES descriptor) is available in two copies. Moreover, *we consider all linear descriptors modulo inversion*. The first assumption is essentially used in research on the intermolecular model, see [54, 55, 59]. The second assumption is quite natural whenever we model double-stranded DNA molecules.

Example 4.2. Consider the micronuclear actin gene in *Sterkiella nova*, see Fig. 3. Denoting each of its MDSs by the pair of incoming/outgoing pointers and/or markers similarly to our previous example, we may write the gene as

$$\delta = (p_3, p_4)(p_4, p_5)(p_6, p_7)(p_5, p_6)(p_7, p_8)(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2)(p_8, p_9).$$

Then δ can be assembled in the intermolecular model as follows:

$$\begin{aligned}
\delta &\xrightarrow{p_5} (p_3, p_4)(p_4, p_5, p_6)(p_7, p_8)(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2)(p_8, p_9) + \cdot p_5(p_6, p_7)\\
&\xrightarrow{p_8} (p_3, p_4)(p_4, p_5, p_6)(p_7, p_8, p_9) + \cdot p_8(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2) + \cdot p_5(p_6, p_7)\\
&\xrightarrow{p_4} (p_3, p_4, p_5, p_6)(p_7, p_8, p_9) + \cdot p_4 + \cdot p_8(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2) + \cdot p_5(p_6, p_7)\\
&\xrightarrow{p_7} (p_3, p_4, p_5, p_6)p_7p_5(p_6, p_7, p_8, p_9) + \cdot p_4 + \cdot p_8(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2)\\
&\xrightarrow{p_6} (p_3, p_4, p_5, p_6, p_7, p_8, p_9) + \cdot p_6p_7p_5 + \cdot p_4 + \cdot p_8(p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2)\\
&\xrightarrow{p_9} (p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2)p_8p_9 + \cdot p_6p_7p_5 + \cdot p_4.
\end{aligned}$$

Assuming that $\overline{\delta}$ is also available, the assembly continues as follows. Here, for a (circular) string τ, we use 2τ to denote $\tau + \tau$:

$$\begin{aligned}
\delta + \overline{\delta} \;\to \ldots \to\; & (p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2)p_8p_9\\
&+ \overline{p_9}\,\overline{p_8}(\overline{p_2}, \overline{b})(p_2, p_3)(\overline{e}, \overline{p_9}, \overline{p_8}, \overline{p_7}, \overline{p_6}, \overline{p_5}, \overline{p_4}, \overline{p_3})\\
&+ 2 \cdot p_6p_7p_5 + 2 \cdot p_4\\
\xrightarrow{p_2}\; & (p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)(\overline{p_3}, \overline{p_2})(b, p_2, p_3)(\overline{e}, \overline{p_9}, \overline{p_8}, \overline{p_7}, \overline{p_6}, \overline{p_5}, \overline{p_4}, \overline{p_3})\\
&+ \overline{p_9}\,\overline{p_8}(\overline{p_2}, \overline{b})p_2p_8p_9 + 2 \cdot p_6p_7p_5 + 2 \cdot p_4\\
\xrightarrow{\overline{p_2}}\; & (p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)(\overline{p_3}, \overline{p_2}, \overline{b})p_2p_8p_9\\
&+ \overline{p_9}\,\overline{p_8p_2}(b, p_2, p_3)(\overline{e}, \overline{p_9}, \overline{p_8}, \overline{p_7}, \overline{p_6}, \overline{p_5}, \overline{p_4}, \overline{p_3})\\
&+ 2 \cdot p_6p_7p_5 + 2 \cdot p_4\\
\xrightarrow{\overline{p_3}}\; & (p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)\overline{p_3}\\
&+ \overline{p_9}\,\overline{p_8p_2}(b, p_2, p_3)(\overline{e}, \overline{p_9}, \overline{p_8}, \overline{p_7}, \overline{p_6}, \overline{p_5}, \overline{p_4}, \overline{p_3}, \overline{p_2}, \overline{b})p_2p_8p_9\\
&+ 2 \cdot p_6p_7p_5 + 2 \cdot p_4\\
\xrightarrow{p_3}\; & p_3(\overline{e}, \overline{p_9}, \overline{p_8}, \overline{p_7}, \overline{p_6}, \overline{p_5}, \overline{p_4}, \overline{p_3}, \overline{p_2}, \overline{b})p_2p_8p_9\\
&+ \overline{p_9}\,\overline{p_8p_2}(b, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, e)\overline{p_3} + 2 \cdot p_6p_7p_5 + 2 \cdot p_4
\end{aligned}$$

$$=2\overline{p_9}\,\overline{p_8p_2}(b,p_2,p_3,p_4,p_5,p_6,p_7,p_8,p_9,e)\overline{p_3}+2\cdot p_6p_7p_5+2\cdot p_4.$$

This calculation shows that the gene is eventually assembled, with p_3 preceding the assembled gene and $p_2p_8p_9$ following it, and that two circular molecules are excised during the assembly.

We refer to Section 5 for intramolecular assembly strategies of the actin I gene in *S. nova* and to Section 6 for a set of properties that are independent of particular assembly strategies, called invariants. The DNA sequences of all molecules produced by gene assembly are particular invariants of the process. We refer to [44] for a detailed discussion on intermolecular assembly strategies and a comparison to intramolecular assemblies.

Consider now contextual versions of the operations. We define rules of the form $(p,x,q)\ (p',x,q')$ with the intended semantics that p (p', resp.) and q (q', resp.) represent contexts flanking x (x', resp.). These contexts are not directly involved in the recombination process, but instead regulate it: recombination may only take place if x (x', resp.) is flanked by p (p', resp.) and q (q', resp.). Our intramolecular operation now becomes $uxvxw \rightarrow \{uxw, \cdot vx\}$, where $u = u'p$,$v = qv' = v''p'$, $w = q'w'$, with similar constraints for the intermolecular operation.

It was shown in [59] that iterated nondeterministic application of these contextual rules to an initial axiom string yields a computing system with the generative power of a Turing machine.

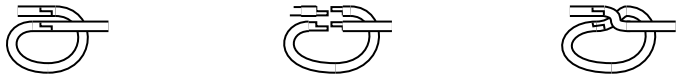
5 The intramolecular model for gene assembly

5.1 Molecular model

The *intramolecular model* was introduced in [33] and [75]. It consists of a set of three irreversible molecular operations explaining the excision of IESs and the unscrambling and ligation of MDSs during the MIC-MAC development. All three operations postulate the folding of a DNA molecule into a specific pattern that allows the alignment of some pointers. Subsequent DNA recombination on those pointers leads to the MDSs (and IESs) being rearranged. We describe each of the three operations in the following.

Loop, direct-repeat excision (in short, ld*)*

The ld *operation* is applicable to a DNA molecule having two occurrences of a pointer, say p, on the same strand. We say in this case that pointer p has a *direct repeat* along the molecule. The molecule is then folded into a *loop* (Fig. 6(a)) so that the two occurrences of p are aligned. Recombination on p is thus facilitated (Fig. 6(b)) and as a result, a linear and a circular molecule are obtained, (Fig. 6(c)). Each of the two molecules has an occurrence of p.

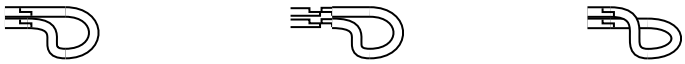


(a) (b) (c)

Fig. 6. Illustration of the ld-rule. (a) The molecule is first folded into a loop, aligning the two direct repeats of a pointer. (b) Recombination is facilitated on the two occurrences of the pointer. (c) One linear and one circular molecule are produced as a result of the operation.

Hairpin, inverted-repeat excision/reinsertion (in short, hi*)*

The hi *operation* is applicable to a DNA molecule having two occurrences of a pointer, say p, on different strands of the molecule. We say in this case that pointer p has an *inverted repeat* along the molecule. Then the molecule is folded into a *hairpin* (Fig. 7(a)) so that the two occurrences of p have a direct alignment. Recombination on p is thus facilitated (Fig. 7(b)), and as a result, a new linear molecule is obtained, where one block of nucleotides has been inverted (Fig. 7(c)).



(a) (b) (c)

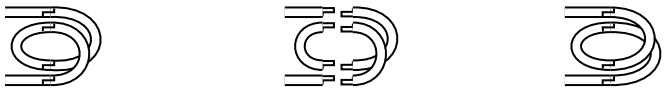
Fig. 7. Illustration of the hi-rule. (a) The molecule is first folded into a hairpin, aligning the two inverted repeats of a pointer. (b) Recombination is facilitated on the two occurrences of the pointer. (c) A new linear molecule is produced as a result of the operation.

Double loop, alternating direct-repeat excision (in short, dlad*)*

The dlad *operation* is applicable to a DNA molecule having two pointers, say p and q, each with two occurrences. All four pointer occurrences should be on the same strand. Moreover, pointer p has one occurrence in-between the two occurrences of q and one occurrence outside them. (By consequence, the same holds true for pointer q with respect to the two occurrences of pointer p.) The molecule is then folded into a double loop (Fig. 8(a)), so that the two occurrences of p and the two occurrences of q are simultaneously aligned. Recombination events on p and on q are facilitated (Fig.8(b)), and as a result, a new linear molecule is obtained, see Fig. 8(c).

Discussion

The $\{\mathsf{ld}, \mathsf{hi}, \mathsf{dlad}\}$ model is often referred to as the *intramolecular model*, to stress that in this model, the input to which operations are applied is always a single molecule.



(a) (b) (c)

Fig. 8. Illustration of the dlad-rule. (a) The molecule is first folded into a double loop, simultaneously aligning two pairs of pointer occurrences. (b) Recombination is facilitated on both pointer alignments. (c) A new linear molecule is produced as a result of the operation.

(In contrast, the model presented in Section 4 is referred to as the *intermolecular model*.) It is important to note however that ld yields as an output two molecules. As such, for the intramolecular assembly to succeed, i.e., to assemble all MDSs, it is essential that all coding blocks remain within one of the molecules produced by ld. This gives two restrictions for applying ld on a pointer p in a successful gene assembly:

(i) No MDSs exist in-between the two occurrences of p. (Equivalently, only one, possibly composite, IES exists in-between the two occurrences of p.) We say in this case that we have a *simple* application of ld. As a result, a (possibly composite) IES is excised as a circular molecule and all MDSs remain on the resulting linear molecule.

(ii) All MDSs are placed in-between the two occurrences of p. We say in this case that we have a *boundary* application of ld. As a result, all MDSs are placed on the resulting circular molecule. The final assembled gene will be a circular molecule. It has been shown in [32] and [29] that using a boundary application of ld can be postponed to the last step of any successful assembly. In this way, in-between the two occurrences of p there is only one (possibly composite) IES, similarly as in the case of simple ld.

The mechanistic details of the alignment and recombination events postulated by the three molecular operations ld, hi and dlad are not indicated in the original proposal for the intramolecular model. Two mechanisms were later proposed in [76] and in [5]. The details of both are discussed in this chapter in Section 7.

5.2 String and graph representations for ciliate genes

We present three different formalizations for the gene assembly process: signed permutations, legal strings and overlap graphs. The first two of these are linear in the sense that the order of the MDSs can be readily seen from the presentation. On the other hand, from the overlap graphs the order of the components is more difficult to capture. The gene assembly process will, however, be different in nature for signed permutations as for legal strings as well as for overlap graphs. Permutations need to be sorted while strings and graphs need to be reduced to empty string and graph, respectively. Another formalization in terms of *descriptors* is given in Section 6.

Signed permutations and legal strings

Each micronuclear arrangement of MDSs and IESs can be represented as a signed permutation over the alphabet $\{M_1, M_2, \ldots, M_\kappa\}$, for an integer $\kappa \geq 1$ corresponding to the number of macronuclear destined sequences that assemble to a functional gene. We identify such a string with a signed permutation over the index set $\{1, 2, \ldots, \kappa\}$.

Example 5.1. Consider the micronuclear arrangement of the actin I gene of *Sterkiella nova*: $M_3M_4M_6M_5M_7M_9\overline{M}_2M_1M_8$. This is represented as the signed permutation $\alpha = 3\,4\,6\,5\,7\,9\,\overline{2}\,1\,8$.

The gene assembly process is equivalent to sorting a signed permutation in proper order, i.e., in the order $p(p+1)\ldots\kappa 1\ldots(p-1)$ or its inverse for suitable p (and κ, the number of MDSs in the micronuclear gene). If $p = 1$ here, then the MDSs are linearly ordered; otherwise they are cyclically ordered.

Representation by strings will preserve the order of the MDSs which are coded as "pairs of pointers".

A string $v \in \Sigma^*$ over an alphabet Σ is said to be a *double occurrence string*, if every letter $a \in \mathsf{dom}(v)$ occurs exactly twice in v. A signing of a nonempty double occurrence string is a *legal string*. A letter $a \in \Sigma \cup \overline{\Sigma}$ is *positive* in a legal string $v \in \Sigma^{\maltese}$, if v contains both a and $\overline{a}$; otherwise, a is *negative* in v.

Example 5.2. Consider the legal string $u = 2\,4\,3\,\overline{2}\,\overline{5}\,3\,4\,5$ of pointers. Pointers 2 and 5 are positive in u, while 3 and 4 are negative in u. On the other hand, the string $w = 2\,4\,3\,\overline{2}\,\overline{5}\,3\,5$ is not legal, since 4 has only one occurrence in w.

Let $u = a_1a_2\ldots a_n \in \Sigma^{\maltese}$ be a legal string over Σ with $a_i \in \Sigma \cup \overline{\Sigma}$ for each i. For $a \in \mathsf{dom}(u)$, let $1 \leq i < j \leq n$ be such that $\|a_i\| = a = \|a_j\|$. Then the substring

$$u_{(a)} = a_i a_{i+1} \ldots a_j$$

is called the *a-interval* of u. Two different letters $a, b \in \Sigma$ are said to *overlap in* u, if the a-interval and the b-interval of u overlap: if $u_{(a)} = a_{i_1} \ldots a_{j_1}$ and $u_{(b)} = a_{i_2} \ldots a_{j_2}$, then either $i_1 < i_2 < j_1 < j_2$ or $i_2 < i_1 < j_2 < j_1$.

Example 5.3. Let $u = 2\,4\,3\,5\,3\,\overline{2}\,\overline{6}\,\overline{5}\,4\,6$ be a signed string of pointers. The 2-interval of u is the substring $u_{(2)} = 2\,4\,3\,5\,3\,\overline{2}$, and hence pointer 2 overlaps with 4 and 5 but not with 3 or 6. Similarly, e.g., $u_{(4)} = 4\,3\,5\,3\,\overline{2}\,\overline{6}\,\overline{5}\,4$ and hence 4 overlaps with 2 and 6.

A signed permutation will be represented by a legal string using the following substitution $\varrho_\kappa \colon \{1, 2, \ldots, \kappa\}^{\maltese} \to \{2, 3, \ldots, \kappa\}^{\maltese}$

$$\varrho_\kappa(1) = 2, \;\; \varrho_\kappa(\kappa) = \kappa, \;\; \varrho_\kappa(p) = p\,(p+1) \quad \text{for } 2 \leq p < \kappa\,,$$

and $\varrho_\kappa(\overline{p}) = \overline{\varrho_\kappa(p)}$ for each p with $1 \leq p \leq \kappa$.

Example 5.4. Consider the signed permutation α from Example 5.1. We have

$$\varrho_9(\alpha) = 34\,45\,67\,56\,78\,9\,\overline{3}\,\overline{2}\,2\,89.$$

Overlap graphs

We turn now to representations by graphs. We use signed graphs to represent the structure of overlaps of letters in legal strings as follows: let $v \in \Sigma^{\maltese}$ be a legal string. The *overlap graph* of v is the signed graph $G_v = (\mathsf{dom}(v), E, \sigma)$ such that

$$\sigma(x) = \begin{cases} +\,, & \text{if } x \text{ is positive in } v\,, \\ -\,, & \text{if } x \text{ is negative in } v\,, \end{cases}$$

and

$$\{x, y\} \in E \iff x \text{ and } y \text{ overlap in } v\,.$$

Example 5.5. Consider the legal string $v = 3\,4\,\overline{5}\,2\,\overline{3}\,\overline{2}\,4\,5$ of pointers. Then its overlap graph G_v is given in Fig. 9.

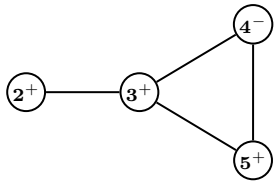


Fig. 9. The overlap graph of the signed string $v = 3\,4\,\overline{5}\,2\,\overline{3}\,\overline{2}\,4\,5$.

Overlap graphs of double occurrence strings are also known as *circle graphs*.

Example 5.6. Notice that the mapping $w \mapsto G_w$ of legal strings to overlap graphs is not injective. The following eight legal strings of pointers have the same overlap graph (of one edge): $2\overline{3}23$, $\overline{3}232$, $232\overline{3}$, $32\overline{3}2$, $\overline{2}\overline{3}23$, $\overline{3}\overline{2}3\overline{2}$, $\overline{2}3\overline{2}\overline{3}$, $3\overline{2}\overline{3}\overline{2}$. For a more complicated example, we mention that the strings $v_1 = 23342554$ and $v_2 = 35242453$ define the same overlap graph.

Reduction graph

Recall that legal strings represent the MIC form of genes. We now introduce a graph, called the *reduction graph*, that represents the MAC form of a gene and the other molecules obtained as results of the assembly, given a legal string (the MIC form of that gene). In this way, the reduction graph represents the end result after recombination on all pointers. First, we define a *2-edge colored graph* as a tuple (V, E_1, E_2, f, s, t), where V are the vertices, $s, t \in V$ are called *source* and *target*, and $f : V\backslash\{s, t\} \to \Gamma$ is a vertex labeling function with Γ a finite set of vertex labels. There are two (not necessarily disjoint) sets of undirected edges E_1 and E_2. We let $\mathsf{dom}(G)$ be the range of f, and say that 2-edge colored graphs G and G' are *isomorphic*, denoted $G \approx G'$, when they are equal up to a renaming of the vertices.

However, we require that the labels of the identified vertices are equal, and that the sources and targets of G and G' are identified.

A reduction graph [10] is a 2-edge colored graph where the two types of edges E_1 and E_2 are called *reality edges* and *desire edges* respectively. Moreover, each vertex, except s and t, is labelled by an element of $\Delta_\kappa = \{2, 3, \ldots, \kappa\}$. As an example, consider the representation of legal string $u = 2\bar{7}473534\bar{2}656$ over $\Pi_\kappa = \Delta_\kappa \cup \overline{\Delta}_\kappa$ given in Fig. 10. We will use this legal string as our running example.

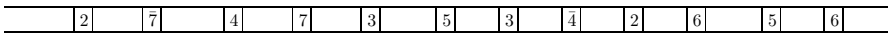


Fig. 10. The representation of $u = 2\bar{7}473534\bar{2}656$.

A reduction graph $\mathcal{R}_u$ for a legal string u is defined in such a way that (1) each occurrence of a pointer of u appears twice (in unbarred form) as a label of a vertex in the graph to represent both sides of the pointer in the representation of u, (2) the reality edges (depicted as 'double edges' to distinguish them from the desire edges) represent the segments between the pointers, (3) the desire edges represent which segments should be glued to each other when recombination operations are applied on the corresponding pointers. To enforce this last requirement, positive pointers are connected by crossing desire edges (cf. pointers 4 and 7 in Fig. 11), while negative pointers are connected by parallel desire edges. The vertices s and t represent the left and right end respectively. Note that, since the reduction graph is fixed for a given u, the end product after recombination is fixed as well. The notion of reduction graph is similar to the breakpoint graph (or reality-and-desire diagram) known from the theory of sorting by reversal, see, e.g., [79] and [70]. A formal definition of reduction graph is found, e.g., in [10]. The reduction graph for string $u = 2\bar{7}473534\bar{2}656$ is in Fig. 11.

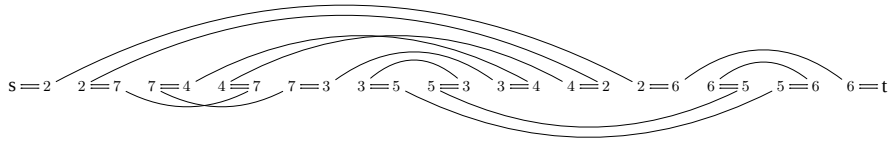


Fig. 11. The reduction graph for $u = 2\bar{7}473534\bar{2}656$.

In depictions of reduction graphs, we will represent the vertices (except for s and t) by their labels, because the exact identities of the vertices are not essential here – we consider reduction graphs up to isomorphism. Note that the reduction graph is defined for the general concept of legal strings. Therefore, the reduction graph represents the end product after recombination of arbitrary sequences of pointers (which by definition come in pairs) – not only those that correspond to sequences of MDSs.

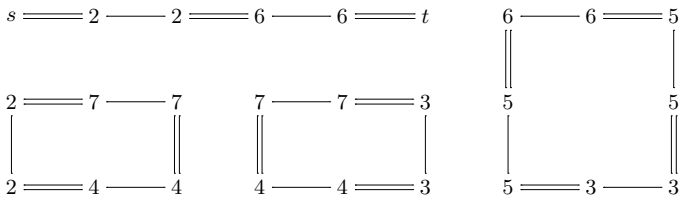


Fig. 12. The reduction graph $\mathcal{R}_u$ of u from the running example.

In the running example, the reduction graph $\mathcal{R}_u$ of u is again depicted in Fig. 12 – we have only rearranged the vertices.

Each reduction graph has a connected component, called the *linear component*, containing both vertices s and t. The other connected components are called *cyclic components*.

5.3 Mathematical formalizations of the intramolecular model

In this section we formalize the gene assembly process involving the three molecular operations ld, hi, and dlad in the framework of legal strings and overlap graphs.

Assembly operations on strings

Recall that each micronuclear MDS structure can be faithfully represented as a signed permutation α, which in turn has a presentation as a legal string $v = \varrho_\kappa(\alpha)$, where κ is the number of the MDSs in the micronuclear gene, and as an overlap graph G_v with κ vertices.

We shall first describe the assembly operations for strings. The rules are (snr) the *string negative rule*, (spr) the *string positive rule*, and (sdr) the *string double rule*. For simplicity we consider only strings of pointers. Recall that we denote $\Delta_\kappa = \{2, 3, \ldots, \kappa\}$ and $\Pi_\kappa = \Delta_\kappa \cup \overline{\Delta}_\kappa$. Below, we assume that $p, q \in \Pi_\kappa$.

- snr_p applies to a legal string of the form $u = u_1 p p u_2$ resulting in

$$\mathsf{snr}_p(u_1 p p u_2) = u_1 u_2 \,. \tag{1}$$

- spr_p applies to a legal string of the form $u = u_1 p u_2 \overline{p} u_3$ resulting in

$$\mathsf{spr}_p(u_1 p u_2 \overline{p} u_3) = u_1 \overline{u}_2 u_3 \,. \tag{2}$$

- $\mathsf{sdr}_{p,q}$ applies to a legal string of the form $u = u_1 p u_2 q u_3 p u_4 q u_5$ resulting in

$$\mathsf{sdr}_{p,q}(u_1 p u_2 q u_3 p u_4 q u_5) = u_1 u_4 u_3 u_2 u_5 \,. \tag{3}$$

We define $\mathsf{dom}(\rho)$ for a string reduction rule ρ by $\mathsf{dom}(\mathsf{snr}_p) = \mathsf{dom}(\mathsf{spr}_p) = \{\|p\|\}$ and $\mathsf{dom}(\mathsf{sdr}_{p,q}) = \{\|p\|, \|q\|\}$ for $p, q \in \Pi_\kappa$.

We adopt the following graphical notations for the applications of these operations:

$$u \xrightarrow{\mathsf{snr}_p} \mathsf{snr}_p(u), \quad u \xrightarrow{\mathsf{spr}_p} \mathsf{spr}_p(u), \quad u \xrightarrow{\mathsf{sdr}_{p,q}} \mathsf{sdr}_{p,q}(u)\,.$$

A composition $\varphi = \varphi_n \dots \varphi_1$ of the above operations φ_i is a *string reduction* of u, if φ is applicable to u. Also, φ is *successful* for u, if $\varphi(u) = \Lambda$, the empty string. Moreover, we define $\mathsf{dom}(\varphi) = \mathsf{dom}(\varphi_1) \cup \mathsf{dom}(\varphi_2) \cup \dots \cup \mathsf{dom}(\varphi_n)$.

Example 5.7. The rule snr_2 is applicable to the legal string $u = 5\,2\,2\,3\,\overline{5}\,\overline{4}\,3\,4$: $\mathsf{snr}_2(u) = 5\,3\,\overline{5}\,\overline{4}\,3\,4$. Moreover, we have

$$5\,2\,2\,3\,\overline{5}\,\overline{4}\,3\,4 \xrightarrow{\mathsf{snr}_2} 5\,3\,\overline{5}\,\overline{4}\,3\,4 \xrightarrow{\mathsf{spr}_{\overline{4}}} 5\,3\,\overline{5}\,\overline{3} \xrightarrow{\mathsf{spr}_3} 5\,5 \xrightarrow{\mathsf{snr}_5} \Lambda,$$

and hence φ is successful for u.

The following is the basic universality result for legal strings.

Theorem 5.8 ([28, 10]). *Each legal string has a successful string reduction.*

Assembly operations on graphs

We shall now describe the assembly operations for graphs. The rules are (gnr) the *graph negative rule*, (gpr) the *graph positive rule*, (gdr) the *graph double rule*.

Let x and y be vertices of a signed graph G.

- gnr_x is applicable to G, if x is isolated and negative. The result is $\mathsf{gnr}_x(G) = G - x$.
- gpr_x is applicable to G, if x is positive. The result is $\mathsf{gpr}_x(G) = \mathsf{loc}_x(G) - x$.
- $\mathsf{gdr}_{x,y}$ is applicable to G, if x and y are adjacent and negative. The result is $\mathsf{gdr}_{x,y}(G) = \mathsf{loc}_x\,\mathsf{loc}_y\,\mathsf{loc}_x(G) - \{x, y\}$ obtained by complementing the edges between the sets $N_G(x) \setminus N_G(y)$, $N_G(y) \setminus N_G(x)$, and $N_G(x) \cap N_G(y)$.

Example 5.9. Consider the overlap graph $G = G_w$ for $w = 3\,\overline{5}\,2\,6\,5\,4\,7\,3\,6\,7\,2\,\overline{4}$; see Fig. 13(a). The graph $\mathsf{gpr}_4(G)$ is given in Fig. 13(b), and the graph $\mathsf{gdr}_{2,3}(G)$ is given in Fig. 13(c).

A composition $\varphi = \varphi_n \dots \varphi_1$ of the above graph operations is called a *graph reduction* for a signed graph G, if φ is applicable to G. Also, φ is *successful*, if $\varphi(G)$ is the empty graph.

Example 5.10. The overlap graph $G = G_w$ given in Fig. 13(a) is reduced to the empty graph by the composition $\mathsf{gpr}_5\,\mathsf{gpr}_6\,\mathsf{gpr}_7\,\mathsf{gpr}_4\,\mathsf{gdr}_{2,3}$.

The above operations are universal for signed graphs:

Theorem 5.11 ([45]). *Each signed graph G has a successful graph reduction.*

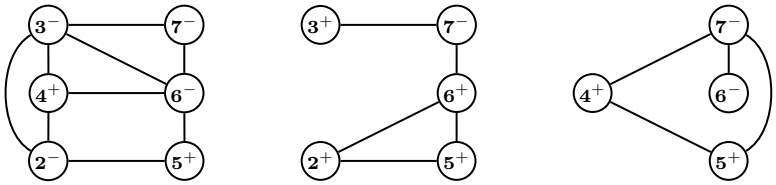


Fig. 13. The graphs (a) $G = G_w$, (b) $\mathsf{gpr}_4(G)$, (c) $\mathsf{gdr}_{2,3}(G)$, where $w = 3\,\overline{5}\,2\,6\,5\,4\,7\;3\,6\;7\,2\;\overline{4}$.

Equivalence of the systems

We study now the relation between the systems for strings and graphs, and we show that there is a correspondence between these operations.

Theorem 5.12 ([30]). *Let w be a legal string. Each string reduction $\varphi = \varphi_n \dots \varphi_1$ for w translates into a graph reduction $\varphi' = \varphi'_n \dots \varphi'_1$ for the overlap graph G_w by the translation:*

$$\mathsf{snr}_p \mapsto \mathsf{gnr}_p, \quad \mathsf{spr}_p \mapsto \mathsf{gpr}_p, \quad \mathsf{sdr}_{p,q} \mapsto \mathsf{gdr}_{p,q}\ .$$

Consequently, if φ is successful for w, then φ' is successful for G_w.

The reverse implication, from graphs to strings, is not as straightforward. Recall first that the mapping from the legal strings to the overlap graphs is not injective.

Denote by $p(w)$ the *first occurrence* of p or $\overline{p}$ in w.

Theorem 5.13 ([30]). *Let w be a legal string, and let φ be a successful reduction for G_w. Then there exists a permutation $\varphi'' = \varphi_n \dots \varphi_1$ of φ, which is successful for G_w, and which can be translated to a successful string reduction $\varphi' = \varphi'_n \dots \varphi'_1$ for w by the following translations:*

$$\mathsf{gnr}_p \mapsto \mathsf{snr}_{p(w)}, \quad \mathsf{gpr}_p \mapsto \mathsf{spr}_{p(w)}, \quad \mathsf{gdr}_{p,q} \mapsto \mathsf{sdr}_{p(w),q(w)}\ .$$

5.4 Properties of intramolecular assemblies

We discuss in this section some properties of intramolecular gene assemblies. Many of these properties hold in all the mathematical models described in Section 5.3. In each case however, we choose to describe the results only on a level that allows for the simplest or the most elegant formulation. For more results we refer to [29, 6]

Nondeterminism and confluence

It is easy to show on all model levels, from the molecular level to that of graphs, that for a given gene (permutation, string, graph, resp.), there can be more than one

strategy to assemble it. Different assembly reduction strategies have recently also been confirmed experimentally [65]. We say that the intramolecular model is *nondeterministic*. Consider, e.g., the signed string associated to the actin I gene in *S. nova*: $u = 3\,4\,4\,5\,6\,7\,5\,6\,7\,8\,9\,\overline{3}\,\overline{2}\,2\,8\,9$. There are 3060 different sequential strategies to reduce this string (assemble the gene), see [29], of which we show in Table 1 only two. Note in particular that these two strategies differ in the number and type of operations used. On the other hand, both strategies are successful, reducing the input string to the empty string. As shown in Theorems 5.8 and 5.11, this is true in general: although several operations may be applicable to a given input, successful strategies for that input exist starting with any of those operations. The biological interpretation is that all (potentially many) assembly strategies of a given micronuclear gene, have the same result: the assembled corresponding macronuclear gene. We call such a model *confluent*. Consequently, ciliates "need not remember" a particular sequential strategy which in turn contributes to the robustness of the gene assembly process.

(a)	(b)
$u_1 = \mathsf{spr}_3(u) = \overline{9}\,\overline{8}\,\overline{7}\,\overline{6}\,\overline{5}\,\overline{4}\,\overline{4}\,\overline{2}\,2\,8\,9$	$u'_1 = \mathsf{snr}_4(u) = 3\,5\,6\,7\,5\,6\,7\,8\,9\,\overline{3}\,\overline{2}\,8\,9$
$u_2 = \mathsf{snr}_4(u_1) = \overline{9}\,\overline{8}\,\overline{7}\,\overline{6}\,\overline{5}\,\overline{2}\,2\,8\,9$	$u'_2 = \mathsf{sdr}_{5,6}(u'_2) = 3\,7\,7\,8\,9\,\overline{3}\,\overline{2}\,8\,9$
$u_3 = \mathsf{spr}_8(u_2) = \overline{9}\,\overline{2}\,2\,5\,6\,7\,5\,6\,7\,9$	$u'_3 = \mathsf{snr}_7(u'_2) = 3\,8\,9\,\overline{3}\,\overline{2}\,8\,9$
$u_4 = \mathsf{spr}_2(u_3) = \overline{9}\,5\,6\,7\,5\,6\,7\,9$	$u'_4 = \mathsf{sdr}_{8,9}(u'_3) = 3\,\overline{3}\,\overline{2}\,2$
$u_5 = \mathsf{sdr}_{5,7}(u_4) = \overline{9}\,6\,6\,9$	$u'_5 = \mathsf{spr}_2(u'_4) = 3\,\overline{3}$
$u_6 = \mathsf{snr}_6(u_5) = \overline{9}\,9$	$u'_6 = \mathsf{spr}_3(u'_5) = \Lambda$
$u_7 = \mathsf{spr}_9(u_6) = \Lambda$	

Table 1. Two reduction strategies for the signed string corresponding to the actin I micronuclear gene in S.nova.

We prove in Section 6 that the assembly strategies of a given gene share a number of other properties beyond yielding the same assembled gene: the number of molecules (linear and circular) produced throughout the assembly, their nucleotide sequence, whether the assembled gene is linear or circular.

The Structure of MAC genes with byproducts

Since legal strings represent the initial configuration (gene in MIC form) and the corresponding reduction graph the end result (the same gene in MAC form and its excised products), it is natural to study the possible forms of reduction graphs. Formally, we characterize now the graphs that are (isomorphic to) reduction graphs.

A graph G isomorphic to a reduction graph must be a 2-edge coloured graph (V, E_1, E_2, f, s, t) such that for each p in the range of f, $p \in \Delta$ and there must be exactly 4 vertices labelled by p. Each vertex must be connected to exactly one (reality) edge from E_1, and each vertex, except s and t, must be connected to exactly one (desire) edge from E_2. Finally, edges from E_2 must connect vertices with a common label. Let us call these graphs *abstract reduction graphs*, and let the set of

abstract reduction graphs be ARG. It turns out that there are graphs in ARG that are not (isomorphic to) reduction graphs.

To obtain a characterization we need one more property of reduction graphs: the ability to linearly order the vertices to resemble its (in general not unique) underlying legal string, as done in Fig. 11. To make this linear order of vertices explicit, we introduce a third set of edges, called *merge edges* , to the reduction graph as done in Fig. 14.

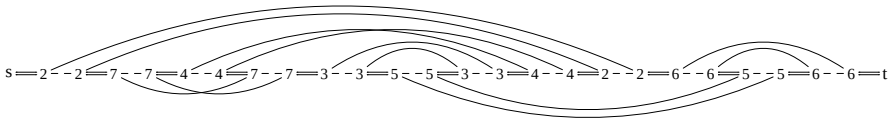


Fig. 14. Merge edges are added to the reduction graph of Fig. 11.

Now, when is a set of edges M for $G \in \mathsf{ARG}$ a set of merge edges? Like desire edges, they have the properties that (1) the edges connect vertices with a common label and (2) each vertex except s and t is connected to exactly one merge edge. Moreover, M and the set E_2 are disjoint – no desire edge is parallel to a merge edge. Finally, the reality edges and merge edges must allow for a path from s to t passing each vertex once. This last requirement is equivalent to the fact that the reality and merge edges induce a connected graph.

If it is possible to add a set of merge edges to the graph, then it is not difficult to see that the graph is isomorphic to a reduction graph $\mathcal{R}_u$. Indeed, we can identify such a u for this reduction graph by simply considering the alternating path from s to t over the reality and merge edges. The orientation (positiveness or negativeness) of each pointer is determined by the crossing or non crossing of the desire edges (exactly as we defined the notion of reduction graph).

To characterize reduction graphs we need the notion of a pointer-component graph. Given an abstract reduction graph, a pointer-component graph describes how the labels of that abstract reduction graph are distributed among its connected components.

Definition 5.14. *Let $G \in \mathsf{ARG}$. The* pointer-component graph *of G, denoted by $\mathcal{PC}_G$, is a multigraph (ζ, E, ε), where ζ is the set of connected components of G, $E = \mathsf{dom}(G)$ and ε is, for $e \in E$, defined by $\varepsilon(e) = \{C \in \zeta \mid C$ contains vertices labelled by $e\}$.*

The pointer-component graph of $G = \mathcal{R}_u$ of Fig. 12 is given in Fig. 15. We have $\zeta = \{C_1, C_2, C_3, R\}$ where R is the linear component and the other elements are cyclic components of $\mathcal{R}_u$.

It is shown in [7] that, surprisingly, $G \in \mathsf{ARG}$ has a set of merge edges precisely when the pointer-component graph $\mathcal{PC}_G$ is a connected graph. In other words:

Theorem 5.15 ([7]). *An abstract reduction graph G is isomorphic to a reduction graph iff $\mathcal{PC}_G$ is a connected graph.*

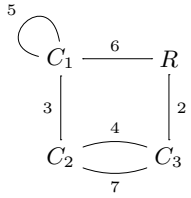


Fig. 15. The graph $\mathcal{PC}_{\mathcal{R}_u}$ of the graph $\mathcal{R}_u$ in Fig. 12.

Consider the legal string $u = 2\bar{7}4735\bar{3}\bar{4}2656$ as before and $v = 2\bar{7}426\bar{5}\bar{3}\bar{4}7356$. It turns out that they have the same reduction graph (up to isomorphism): $\mathcal{R}_u \approx \mathcal{R}_v$ (see Fig. 11). The reason for this is that a reduction graph may have more than one set of merge edges – each one corresponding to a different legal string. Thus, there can be many legal strings giving the same reduction graph. In [7] it is shown how for a given legal string u we can obtain precisely the set of all legal strings having the same reduction graph (up to isomorphism). In fact, it turns out that this set is exactly the set of all legal strings obtained by applying compositions of the following string rewriting rules.

For all $p, q \in \Pi_\kappa$ with $\|p\| \neq \|q\|$ we define

- the *dual string positive rule* for p is defined by $\mathsf{dspr}_p(u_1 p u_2 p u_3) = u_1 p \bar{u}_2 p u_3$,
- the *dual string double rule* for p, q is defined by $\mathsf{dsdr}_{p,q}(u_1 p u_2 q u_3 \bar{p} u_4 \bar{q} u_5) = u_1 p u_4 q u_3 \bar{p} u_2 \bar{q} u_5$,

where $u_1, u_2, \ldots, u_5$ are arbitrary (possibly empty) strings over Π_κ. Notice the strong similarities of these rules with the string positive rule and string double rule. As an example, if we take $u = 2\bar{7}4735\bar{3}\bar{4}2656$ and $v = 2\bar{7}426\bar{5}\bar{3}\bar{4}7356$ given earlier, then $\mathsf{dsdr}_{4,\bar{5}}\ \mathsf{dspr}_3(u) = v$ and hence both legal strings indeed have a common reduction graph.

Intermediate legal strings

We now show that we can generalize the notion of reduction graph to allow for representations of any intermediate product during the reduction process. In such an intermediate product some pointers, represented as a subset D of $\mathsf{dom}(u)$, where u is a legal string, have not yet been used in recombination operations, while the other pointers, in $\mathsf{dom}(u) \backslash D$, have already been used in recombination operations. A reduction graph of u with respect to this set D, denoted by $\mathcal{R}_{u,D}$, represents such intermediate product. As before, we simply ignore the pointers in D – they are put as strings on the reality edges which are now directed edges. Fig. 16 gives an example of $\mathcal{R}_{u,D}$ with $u = 2\bar{7}4735\bar{3}\bar{4}2656$ and $D = \{2, 4\}$ (recall that Λ represents the empty string).

We denote the legal string obtained from a legal string u by removing the pointers from $D \subseteq \mathsf{dom}(u)$ and its barred variants by $\mathsf{rem}_D(u)$. In our example, $\mathsf{rem}_D(u) = \bar{7}7353656$. We define $\mathsf{red}(u, D)$ as the label of the alternating path from s to t. Thus

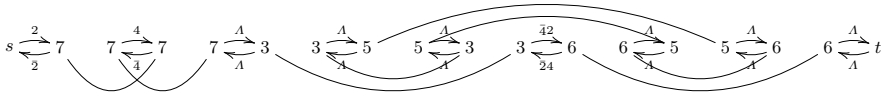


Fig. 16. Graph $\mathcal{R}_{u,D}$ with $u = 2\bar{7}47353\bar{4}2656$ and $D = \{2, 4\}$.

in our example $\mathsf{red}(u, D) = 2\bar{4}\bar{4}2$. Assuming that gene assembly is intramolecular (all recombination takes place on a single DNA molecule), then the cyclic connected components must have only empty strings as edge labels. In our example it is easy to obtain an 'invalid' intermediate product: take, e.g., $D = \{3, 4, 5, 6, 7\}$. Hence, it is not possible to first recombine pointer 2, followed by recombination of the remaining pointers.

Theorem 5.16 ([10]). *Let u be a legal string, let φ be a composition of reduction rules with $\mathsf{dom}(\varphi) \subseteq \mathsf{dom}(u)$, and let $D = \mathsf{dom}(u) \setminus \mathsf{dom}(\varphi)$. Then φ is applicable to u iff φ is applicable to $\mathsf{rem}_D(u)$ and $\mathsf{red}(u, D)$ is a legal string with domain D. Moreover, if this is the case, then $\varphi(u) = \mathsf{red}(u, D)$.*

As a consequence of Theorem 5.16, reductions φ_1 and φ_2 with the same domain have the same effect: $\varphi_1(u) = \varphi_2(u)$ for all legal strings u. Note that in general there are $D \subseteq \mathsf{dom}(u)$ for which there is no reduction φ of u with $D = \mathsf{dom}(\varphi(u))$. In our example $\mathsf{red}(u, D)$ is a legal string with domain D and we have, e.g., $(\mathsf{snr}_6\ \mathsf{sdr}_{3,5}\ \mathsf{spr}_{\bar{7}})(u) = 2\bar{4}\bar{4}2 = \mathsf{red}(u, D)$.

Cyclic components

Since the reduction graph is a representation of the end result after recombination, the cyclic components of a reduction graph represent circular molecules. If we now consider again the intramolecular model of gene assembly, we notice that each such molecule is obtained by loop recombination. Hence, although there can be many different sequences of operations that obtain the fixed end product, the *number* of loop recombination operations (string negative rules in the model) in each such sequence is the same.

Theorem 5.17 ([10]). *Let N be the number of cyclic components in the reduction graph of legal string u. Then every successful reduction of u has exactly N string negative rules.*

Example 5.18. Since $\mathcal{R}_u$ in Fig. 12 has three cyclic components, by Theorem 5.17, every successful reduction φ of u has exactly three string negative rules. For example $\varphi = \mathsf{snr}_2\ \mathsf{snr}_{\bar{4}}\ \mathsf{spr}_{\bar{7}}\ \mathsf{snr}_6\ \mathsf{sdr}_{3,5}$ is a successful reduction of u. Indeed, φ has exactly three string negative rules. Alternatively, $\mathsf{snr}_6\ \mathsf{snr}_3\ \mathsf{snr}_7\ \mathsf{spr}_2\ \mathsf{spr}_{\bar{5}}\ \mathsf{spr}_4$ is also a successful reduction of u, with a different number of (spr and sdr) operations.

It turns out that the reduction graph also allows for determining *on which pointers* the string negative rules can be applied using the pointer-component graph [9]. For

convenience, let us denote $\mathcal{PC}_{\mathcal{R}_u}$ by $\mathcal{PC}_u$. Also, let us denote $\mathcal{PC}_u|_D$ as the graph obtained from $\mathcal{PC}_u$ by removing the edges outside D. Finally, for a reduction φ, let $\mathsf{snrdom}(\varphi) \subseteq \mathsf{dom}(\varphi)$ be the (unbarred) pointers used in snr rules in φ.

Theorem 5.19 ([9]). *Let u be a legal string, and let $D \subseteq \mathsf{dom}(u)$. There is a successful reduction φ of u with $\mathsf{snrdom}(\varphi) = D$ iff $\mathcal{PC}_u|_D$ is a tree.*

In our running example, we see that $D = \{2, 3, 6\}$ induces a (spanning) tree of $\mathcal{PC}_u$. Therefore there is a successful reduction φ of u with $\mathsf{snrdom}(\varphi) = D$. Indeed, we have $\mathsf{sdr}_{\bar{4},5}\ \mathsf{spr}_{\bar{7}}(u) = 226336$. It is clear that we can extend $\mathsf{sdr}_{\bar{4},5}\ \mathsf{spr}_{\bar{7}}$ to a successful reduction which applies string negative rules on 2, 3, and 6. Notice that here snr_3 must be applied *before* snr_6. In fact in [9] it is shown that the possible orders in which the string negative rules can be applied is also deducible from $\mathcal{PC}_u$ by considering rooted trees.

The results above can be carried over to intermediate products; e.g., the number of string negative rules from u to $\varphi(u)$ is fixed and is equal to the number of cyclic components of $\mathcal{R}_{u,D}$.

5.5 Simple and parallel gene assemblies

The general formulation of the intramolecular operations allows for the aligned pointers to be arbitrarily far from each other. We discuss in this section a *simple* variant of the model, where all alignments and folds involved in the operations are *local*. In the simple versions of ld, hi, and dlad the pointers involved in the recombination are at a minimal distance from each other. It turns out that the simple model is able to explain the successful assemblies of all currently known micronuclear gene sequences, see [11, 75, 63]. We discuss in this section the molecular and the mathematical formulation of the simple model and indicate some interesting properties of the model.

In the second part of this section we discuss a notion of *parallel gene assembly*. In each (parallel) step of the assembly we apply a number of well-selected operations simultaneously in such a way the the total number of steps is minimal. In each step the operations are selected in such a way that their application is independent of the others applied in the same step: all sequential compositions of those operations are applicable to the current graph. Several difficult computational problems arise in this context, including deciding whether a given graph has a parallel assembly of a given length, or deciding whether there are graphs (or even trees) of arbitrarily high parallel complexity.

Simple gene assembly

The three intramolecular operations allow in their general formulation that the MDSs participating in an operation may be located anywhere along the molecule. Arguing on the principle of parsimony, a simplified model was discussed already in [75] and then formalized in [46], asking that all operations are applied ‘locally’. In the simple

model the restriction is 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. In one of them, which we refer to as the *simple model* [61], both micronuclear, as well as composite MDSs (obtained by splicing of several micronuclear MDSs) may be manipulated in each of the three molecular operations. In the other, called the *elementary model* and introduced in [42, 43], 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. We discuss in this section only the simple model and refer for details of the elementary model to [42, 43, 63, 69].

We already discussed in Section 5.1 that ld must always be *simple* in a successful assembly. As such, the effect of ld is that it will combine two consecutive MDSs into a bigger composite MDS. For example, consider that M_3M_4 is a part of the molecule, i.e., MDS M_4 succeeds M_3 being separated by one IES I. Thus, pointer 4 has two occurrences that flank I: one in the end of MDS M_3 and the other one in the beginning of MDS M_4. Then ld makes a fold as in Fig. 6(a) aligned by pointer 4, IES I is excised as a circular molecule and M_3 and M_4 are combined into a longer coding block as shown in Fig. 6(c).

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

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, see Fig. 17(a).

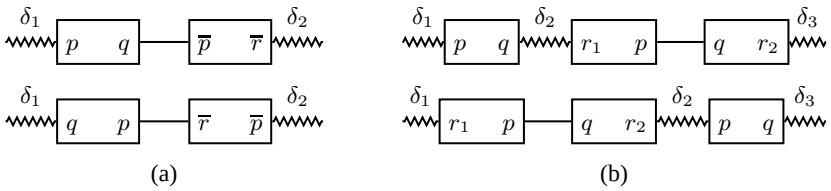


Fig. 17. The MDS/IES structures where (a) *simple hi*-rules and (b) *simple dlad*-rules are applicable. The MDSs are indicated by rectangles and their flanking pointers are shown. Between the two MDSs there is only one IES represented by a straight line. [46, 75]

An application of dlad on pointers p, q is *simple* if the sequence between the first occurrences of p and q, as well as the sequence between the second occurrences of p and 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, see Fig. 17(b).

The simple operations can be formalized as operations on signed permutations, signed strings and signed graphs. We only give here the definitions for the string-based operations, where the mathematical formulation is more concise. For the other formulations, including the relationships among these models, we refer to [46, 63, 8].

The *simple* hi *operation* for pointer p, denoted sspr_p, is applicable to strings of the form $u = u_1 p u_2 \overline{p} u_3$, where $|u_2| \leq 1$, resulting in $\mathsf{sspr}_p(u_1 p u_2 \overline{p} u_3) = u_1 \overline{u_2} u_3$.

The *simple* dlad *operation* for pointers p, q, denoted $\mathsf{ssdr}_{p,q}$, is applicable to strings of the form $u = u_1 p q u_2 p q u_3$, resulting in $\mathsf{ssdr}_{p,q}(u_1 p q u_2 p q u_3) = u_1 u_2 u_3$.

Let ϕ be a composition of snr, sspr, and ssdr operations such that ϕ is applicable to string u. We say that ϕ is a *simple reduction* for u if either $\phi(u) = \Lambda$ (in which case we say that ϕ is *successful*), or $\phi(u) \neq \Lambda$ and no simple operation is applicable to $\phi(u)$ (in which case we say that ϕ is *unsuccessful*). For example, the string reduction in Table 1(b) is simple, unlike the one in Table 1(a).

The simple model has a number of properties that do not hold for the general model. One of them concerns the length of reduction strategies for a given string. While in the general model a string may have reduction strategies of different lengths, see the example in Table 1, the same is not true in the simple model, see the next result of [62]. Moreover, if one considers *parallel applications of simple operations* (a notion of [62] that is not defined in this chapter), we get a new twist: for any given n there exists a string having maximal parallel reductions of any length between n and $2n$.

Theorem 5.20 ([62]). *Let u be a signed double occurrence string and ϕ, ψ two reduction strategies for u. Then ϕ and ψ have the same number of operations.*

Regarding the outcome of reduction strategies, the simple model is different than the general model in several respects; e.g., there are strings that cannot be reduced in the simple model, unlike in the general model where all strings have reduction strategies. Indeed, no simple operation is applicable to the string $\overline{2}\,4\,3\,4\,2\,3$. The following is a result of [61].

Theorem 5.21 ([61]). *No signed string has both successful and unsuccessful reductions in the simple model.*

On the other hand, the outcome of various strategies for a given string can differ; e.g., for $u = 2\,3\,4\,6\,7\,8\,5\,6\,7\,8\,2\,3\,4\,5$, $u_1 = \mathsf{ssdr}_{2,3} \circ \mathsf{ssdr}_{7,8}(u) = 4\,6\,5\,6\,4\,5$, whereas $u_2 = \mathsf{ssdr}_{3,4} \circ \mathsf{ssdr}_{6,7}(u) = 2\,8\,5\,8\,2\,5$. The strings u_1 and u_2 are, however, identical modulo a relabeling of their letters. This observation can be extended to define a notion of *structure* that can be used to prove that the results of various reductions, although different, always have the same structure. For details, we refer to [61], where the discussion is in terms of signed permutations, rather than signed strings.

Parallel gene assembly

The notion of parallelism is usually defined in concurrency theory for processes whose application is independent of each other. In other words, a number of processes can be applied in parallel to a signed graph if they can be (sequentially) applied in any order. Adopting this approach, the following gives the definition of parallel application of the three molecular operations on a signed graph. For a similar discussion, albeit technically more tedious, on the level of signed strings, we refer to [40].

Definition 5.22 ([40]). *Let S be a set of k* gnr, gpr, *and* gdr *operations and let G be a signed graph. We say that the rules in S* are applicable in parallel *to G if for any ordering $\varphi_1, \varphi_2, \ldots, \varphi_k$ of S, the composition $\varphi_k \circ \cdots \circ \varphi_1$ is applicable to G.*

Based on the definition of parallelism, which presumes that the rules are applicable in any possible order, the following theorem shows that the result is always the same regardless of the order in which they are applied.

Theorem 5.23 ([40]). *Let G be a signed graph and let S be a set of operations applicable in parallel to G. Then for any two compositions φ and ψ of the operations of S, $\varphi(G) = \psi(G)$.*

Based on Theorem 5.23, we can write $S(G) = \varphi(G)$ for any set S of operations applicable in parallel to G and any composition φ of these operations. We define the notion of parallel complexity as follows.

Definition 5.24 ([41]). *Let G be a signed graph, and let $S_1, \ldots, S_k$ be sets of* gnr, gpr, gdr *operations. If $(S_k \circ \ldots \circ S_1)(G) = \varnothing$, then we say that $S = S_k \circ \ldots \circ S_1$ is a* parallel reduction *for G. In this case the* parallel complexity *of S is $\mathcal{C}(S) = k$. The* parallel complexity *of the signed graph G is:*

$$\mathcal{C}(G) = \min\{\mathcal{C}(S) \mid S \text{ is a parallel reduction strategy for } G\}.$$

Deciding whether a given set of graph operations is applicable in parallel to a given graph turns out to be a difficult problem if gdr operations are involved. When at most two gdr operations are involved, then simple characterizations were given in [39]. The computational complexity of the general problem was upper-bounded in the co-NP class, see [2].

It can be easily verified that the parallel complexity of the graphs corresponding to the currently known micronuclear gene sequences is at most two, see [38]. However, examples of graphs of higher complexity can be given. For example, the graph with the highest known complexity has 24 vertices and can be reduced in 6 parallel steps. The tree with the highest known parallel complexity has 12 vertices and can be reduced in 5 parallel steps. We refer to [38] for more examples. Although the parallel complexity of certain types of graphs (e.g., for uniformly signed trees) is known to be finitely bounded, see [39], the general problem is currently open and seems to be very difficult. In particular, it seems to require a characterization for the parallel applicability of arbitrary sets of operations, another open problem. The problem is open

even in seemingly simpler cases: for trees, or for negative graphs. The computational complexity of deciding whether the parallel complexity of a given graph is upper bounded by a given contact was placed in the $\mathsf{NP}^{\mathsf{NP}}$ class in [2]. Two algorithms for computing the parallel complexity of signed graphs were given in [1] and [2], both with exponential computational complexity. A visual graph editor including support for computing the parallel complexity of signed graphs can be found in [68].

5.6 Gene assembly by folding and unfolding

In this section we consider gene assembly from a somewhat more general viewpoint. The section is based on [31]. The molecular operations provided by the gene assembly process each involve one molecule. This observation underlies the model of gene assembly as fold-and-recombine computing paradigm. For convenience, we consider circular graphs to be representations of DNA molecules. Our initial situation is a set of circular DNA molecules represented by bicolored and labeled circular graphs. The fold-and-recombine process is reflected by a two-stage processing of the graphs: (1) fold on vertices representing pointers; (2) unfold using a pairing function. In this setup, gene assembly becomes a dynamic process for recombination graphs.

We allow graphs with multiple edges and loops. The vertices denote all pointers of the gene. We consider bicolored graphs, where color 1 is used to indicate an IES and color 2 is used to indicate an MDS. To represent the sequence of nucleotides comprising various (IES or MDS) segments we use a labeling of the edges.

Each edge will be oriented in both directions: $a = (x, y)$ and $\overline{a} = (y, x)$ are *reverse pairs*. Let V be a set of vertices, and consider E as a set of edge symbols such that $E = \{e_1, \ldots, e_n, \overline{e}_1, \ldots, \overline{e}_n\}$ with $\overline{\overline{e}}_i = e_i$. A *(general) bicolored graph* G consists of an *end point map* $\varepsilon_G = \varepsilon \colon E \to V \times V$ such that $\varepsilon(\overline{e}) = \overline{\varepsilon(e)}$ for all $e \in E$; a *labeling* $f_G = f \colon E \to \Sigma^{\maltese}$ for an alphabet Σ, with $f(\overline{e}) = \overline{f(e)}$ for all $e \in E$; a *coloring* $h_G = h \colon E \to \{1, 2\}$ such that $h(e) = h(\overline{e})$ for all $e \in E$.

For simplicity, we write $e = (x, y)$ for $\varepsilon(e) = (x, y)$. An edge $e = (x, y) \in E$ is a *loop*, if $x = y$. The *valency* $\mathrm{val}_G(x)$ of a vertex x is the number of edges leaving x. A bicolored graph is *even* if its valencies are all even. A bicolored graph G is a *recombination graph*, if $\mathrm{val}_G(x) \in \{2, 4\}$ for all x, and every vertex of valency 4 is balanced: two incident edges have color 1 and the other two have color 2; see Fig. 18(a).

For each vertex x in an even bicolored graph G let ψ_x be a bijection that maps incoming edges to outgoing edges respecting inversions, such that $\psi_x(\overline{\psi_x(e)}) = \overline{e}$ and $\psi_x(e) = \overline{e}$ if and only if e is a loop. Then the map $\psi \colon x \mapsto \psi_x$ is a *pairing*. Each recombination graph G has the *natural pairing* ψ where e and $\psi_x(e)$ have the same color whenever $\mathrm{val}_G(x) = 4$.

Folding and unfolding

A pair $p = \{x, y\} \in E(V)$ will be called a *pointer* with *ends* x and y. A set P of mutually disjoint pointers is a *pointer set*. The *p-folded graph* $G * p$ is obtained by identifying the ends of p; see Fig. 18(a) and (b). For a pointer set $\{p, q\}$, $G * p * q =$

$G * q * p$. This allows us to define, for a pointer set $P = \{p_1, \ldots, p_m\}$, the *P-folded graph* $G * P$ as $G * p_1 * \ldots * p_m$.

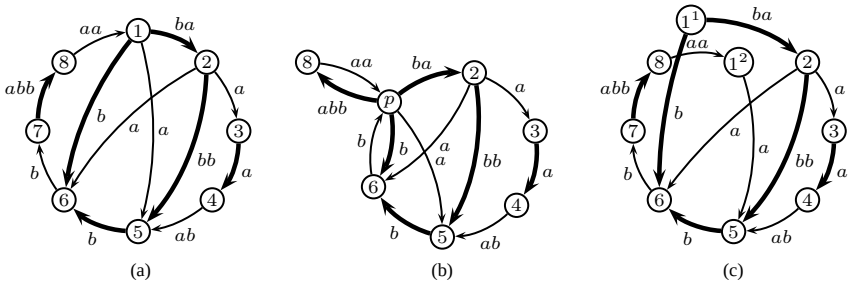


Fig. 18. (a) A recombination graph, where color 1 is represented by thick edge; (b) The p-folded graph $G * p$ for $p = \{1, 7\}$. (c) The ψ-unfolded graph $G \diamond_\psi 1$ with the natural pairing. [31]

Let G be an even bicolored graph with a pairing ψ. For a vertex x, let $e_{11}, e_{12}, \ldots$, e_{m1}, e_{m2} be the incoming edges with $\psi_x(e_{i1}) = \overline{e}_{i2}$. In the *$\psi$-unfolded graph* the vertex x is replaced by the new vertices $x^1, \ldots, x^m$ and the edges are redirected according to the pairing ψ_x; see Fig. 18(c), where the redirection is determined by the colors.

Notice that if G is a recombination graph, so is $G \diamond_\psi x$. Also, if $x \neq y$, then $G \diamond_\psi x \diamond_\psi y = G \diamond_\psi y \diamond_\psi x$. Therefore, we can write $G \diamond_\psi A = G \diamond_\psi x_1 \diamond_\psi \ldots \diamond_\psi x_m$ for a subset $A = \{x_1, \ldots, x_m\}$.

For an even bicolored graph G with a pairing ψ, let $F(G) = \{x \in V_G \mid \mathrm{val}_G(x) \geq 4\}$. Then the graph $G \diamond_\psi F(G)$ is called the *ψ-unfolded graph* of G.

Lemma 5.25 ([31]). *If G is an even bicolored graph with a pairing ψ, then its ψ-unfolded graph is a disjoint union of cycles.*

Let G be a bicolored graph with a pointer set P, and let ψ be a pairing of the P-folded graph $G * P$. We denote $G \circledast_\psi P = (G * P) \diamond_\psi P$. We shall write $G \circledast P$ for $G \circledast_\psi P$, if $G * P$ is a recombination graph and ψ is its natural pairing.

Lemma 5.26 ([31]). *Let G be a disjoint union of bicolored cyclic graphs. Let P be a pointer set of G, and let ψ be a pairing of $G * P$. Then $G \circledast_\psi P$ is a disjoint union of bicolored cyclic graphs.*

Assembled graphs of genomes

Let G be a bicoloured cyclic graph with $V_G = \{x_1, \ldots, x_n\}$ and the edge set $E_G = \{e_1, \ldots, e_n, \overline{e}_1, \ldots, \overline{e}_n\}$, where $e_i = (x_i, x_{i+1})$ and $x_{n+1} = x_1$. A vertex x_i is a *boundary vertex* of G, if $h_G(e_{i-1}) \neq h_G(e_i)$, where $i - 1$ is modulo n. A path π is a *segment*, if its edges have color 1 and the ends of π are boundary vertices. For

a disjoint union $G = \sum_{i=1}^{m} G_i$ of bicolored cyclic graphs G_i, we let its boundary vertex set be the union of corresponding sets of the components.

A pair $\mathcal{G} = (G, P)$ is called a *genome*, if G is a disjoint union of bicolored cyclic graphs and P is a pointer set of G containing boundary vertices only; see Fig. 19.

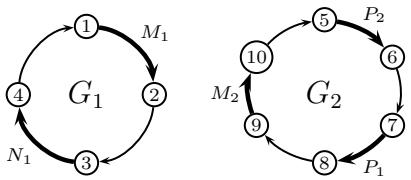


Fig. 19. The genome (G, P) with $P = \{p, q\}$ for $p = \{2, 9\}$ and $q = \{5, 8\}$. The labels correspond to the MDSs M_1, M_2, N_1 and P_1, P_2. The labels corresponding to IESs are omitted. [31]

The *assembled genome* of a genome $\mathcal{G} = (G, P)$ is $A(\mathcal{G}) = (G \circledast P, \varnothing)$, and it is a genome. Each segment of the unfolded graph $\gamma \circledast P$ is a *gene*.

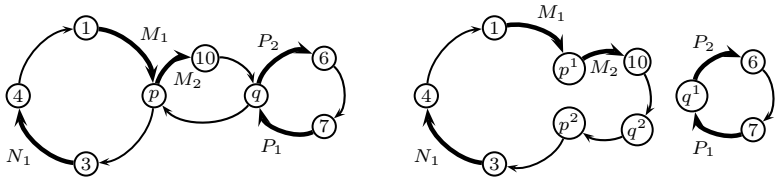


Fig. 20. Let $\mathcal{G}$ be the genome of Fig. 19. Then $G * P$ is given in (a). Unfolding gives $G \circledast P$ in (b). The genes of $\mathcal{G}$ are $g_1 \colon 1 \to p^1 \to 10$ (with the value $M_1 M_2$), $g_2 \colon 7 \to q^1 \to 6$ (with the value $P_1 P_2$), and $g_3 \colon 3 \to 4$ (with the value N_1).[31]

For $\mathcal{G} = (G, P)$, a sequence $\mathcal{S} = (P_1, P_2, \ldots, P_m)$ of subsets of P is an *assembly strategy* of $\mathcal{G}$, if $\{P_1, \ldots, P_m\}$ is a partition of P. One can show that in a genome $\mathcal{G} = (G, P)$, if $P_1, P_2 \subseteq P$ are disjoint, then $(G \circledast P_1) \circledast P_2 = G \circledast (P_1 \cup P_2) = (G \circledast P_2) \circledast P_1$. This gives the following general invariance property:

Theorem 5.27 ([31]). *Every assembly strategy $\mathcal{S} = (P_1, P_2, \ldots, P_m)$ of a genome produces the same assembled genome $\mathcal{G} = (G, P)$: $G \circledast P = G \circledast P_1 \circledast P_2 \circledast \ldots \circledast P_m$.*

A pointer set $R \subseteq P$ of $\mathcal{G} = (G, P)$ is *intracyclic*, if any two parts g' and g'' of each gene g that lie in the same connected component of G, lie in the same connected component of $G \circledast R$.

Theorem 5.28 ([31]). *For each genome $\mathcal{G}$, there exists an intracyclic assembly strategy $\mathcal{S} = (P_1, P_2, \ldots, P_m)$ such that $1 \leq |P_i| \leq 2$ for all i.*

We can also show:

Theorem 5.29 ([31]). *Let $\mathcal{G} = (G, P)$ be a genome for a connected $G \circledast P$. Then there is a genome $\mathcal{G}' = (G', P')$, where G' is connected, such that $A(\mathcal{G}) = A(\mathcal{G}')$, and $\mathcal{G}'$ has an assembly strategy $\mathcal{S} = (P_1, P_2, \ldots, P_m)$ of $\mathcal{G}'$ for which $1 \leq |P_i| \leq 2$ for all i, and each $G \circledast \cup_{i=1}^{j} P_i$ is a cyclic graph for each j.*

6 Invariant properties of gene assembly

As discussed already in this chapter, both an intermolecular model and an intramolecular model exist for gene assembly. Moreover, both models are nondeterministic: for a given gene there may be several assembly strategies and also, a gene may be assembled either on a linear, or on a circular molecule. As such, a natural question is that of *invariants*: what properties of the assembled gene and of the assembly process hold for all assembly strategies of both models? For example, for a given gene, is the set of molecules excised during the assembly an invariant of the process? The same question for whether or not the assembled gene is linear of cyclic, and for whether or not the obtained structure of the IES's is fixed for given gene is also open.

An affirmative answer to these questions was given already in [32] for the intramolecular model, showing that these properties are invariants of the intramolecular model. We follow here a presentation of [67], where the result is given in a stronger form, showing that the properties above are invariants of *any model based on the paradigm of pointer-directed assembly*. This result may also be deduced based on the graph-theoretical framework of [31]. The presentation we give in this section is in terms of strings and permutations. We refer to [67] for more details, examples and full proofs.

6.1 Gene structure

We introduce in this section a novel formal representation for the gene structures of ciliates, able to track the transformations witnessed by a gene from its micronuclear form to its assembled form. We first represent a gene as a signed permutation over the alphabet of MDSs by denoting their sequence and orientation. We then extend our notation to denote also the IESs and all the pointers.

We recall that $u, v \in \Sigma^{\pm}$ are called *equivalent*, denoted $u \approx v$, if u is a conjugate of either v or $\overline{v}$. Two finite sets $X_1, X_2 \subseteq \Sigma^{\pm}$ are called *equivalent*, denoted $X_1 \approx X_2$, if they have the same number of elements and for any $x_i \in X_i$ there is $x_j \in X_j$ such that $x_i \approx x_j$, with $i, j = 1, 2$, $i \neq j$. Intuitively, if u denotes a circular DNA molecule and $u \approx v$, then v denotes the same molecule, potentially starting from a different nucleotide and/or in the reverse direction. Similarly, if X_1 denotes a set of circular molecules and $X_1 \approx X_2$, then X_2 denotes the same set of molecules.

We denote the MDSs of the given gene by letters from the alphabet $\mathcal{M}_n = \{M_1, M_2, \ldots, M_n\}$ in the order they occur in the macronuclear gene, where $n \geq 1$. Thus, the sequence of MDSs in the macronuclear gene is $M_1 M_2 \ldots M_n$. On the other hand, the sequence of MDSs in the micronuclear gene is in general a signed permutation over $\mathcal{M}$.

Example 6.1. The MDSs of the micronuclear gene actin I in *S. nova may be represented as the signed permutation* $M_3M_4M_6M_5\ M_7M_9\overline{M}_2M_1M_8$*, see [74]. In this gene, MDS* M_2 *is inverted.*

We call $\mathcal{M}_n$-*descriptor* any signed permutation over $\mathcal{M}_n$. We say that μ is an *assembled* $\mathcal{M}_n$-*descriptor* if μ or $\overline{\mu}$ are of the form $M_i \dots M_n M_1 \dots M_{i-1}$, for some $1 \leq i \leq n$.

Consider now the alphabet of IESs $\mathcal{I}_n = \{I_0, I_1, \dots, I_n\}$. For any $J \subseteq \mathcal{M}_n \cup \mathcal{I}_n$, a signed permutation over J will be called an $\mathcal{MI}_n$-*descriptor*.

Let π_n be the projection $\pi_n : (\mathcal{M}_n \cup \mathcal{I}_n)^{\maltese} \to \mathcal{M}_n^{\maltese}$. We say that $\delta \in (\mathcal{M}_n \cup \mathcal{I}_n)^{\maltese}$ is an *assembled* $\mathcal{MI}_n$-descriptor if $\pi_n(\delta)$ is an assembled $\mathcal{M}_n$-descriptor.

We can associate an $\mathcal{MI}_n$-descriptor to any $\mathcal{M}_n$-descriptor as follows. Let $\mu = \widetilde{M}_{i_1}\widetilde{M}_{i_2} \dots \widetilde{M}_{i_n}$ be an $\mathcal{M}_n$-descriptor, where $\widetilde{M}_{i_k} \in \{M_{i_k}, \overline{M}_{i_k}\}$ and $i_1, i_2, \dots, i_n$ is a permutation over $\{1, 2, \dots, n\}$. Then the $\mathcal{MI}_n$-*descriptor associated to* μ is $\tau_\mu = I_0\widetilde{M}_{i_1}I_1\widetilde{M}_{i_2}I_2 \dots \widetilde{M}_{i_n}I_n$ – we denote by $I_0, I_1, \dots, I_n$ the non-coding blocks separating the MDSs. We say in this case that τ_μ is a *micronuclear* $\mathcal{MI}_n$-*descriptor*.

Example 6.2. (i) The $\mathcal{MI}_9$-descriptor associated to the actin I gene in S.nova, see Example 6.1, is $I_0M_3I_1M_4I_2M_6I_3\ M_5I_4M_7I_5M_9I_6\overline{M}_2I_7M_1I_8M_8I_9$.
(ii) $\delta = I_0\overline{I}_3\overline{M}_3\overline{M}_2\overline{M}_1I_2I_1$ is an assembled $\mathcal{M}_3$-descriptor.

We extend now the $\mathcal{MI}_n$-descriptors to include also the information about the position of pointers in the gene. Consider then the alphabet $\mathcal{P}_n = \{2, 3, \dots, n\}$ and denote the markers by b and e. Denote $\Sigma_n = \mathcal{M}_n \cup \mathcal{I}_n \cup \mathcal{P}_n \cup \{b, e\}$ and let π_n be the projection $\pi_n : \Sigma_n^{\maltese} \to (\mathcal{M}_n \cup \mathcal{I}_n)^{\maltese}$. We say that $\sigma \in \Sigma_n^{\maltese}$ is a Σ_n-*descriptor* if it has one of the following forms:

(i) $\sigma = \alpha_0 p_1 p_1 \alpha_1 p_2 p_2 \dots p_k p_k \alpha_k$, $\alpha_0\alpha_k \neq \Lambda$, or
(ii) $\sigma = p_1\alpha_1 p_2 p_2 \dots p_{k-1}p_{k-1}\alpha_k p_1$,

where $k \geq 0$, $p_i \in \mathcal{P}_n \cup \overline{\mathcal{P}}_n$, $\alpha_i \in (\Sigma_n \setminus \mathcal{P}_n)^{\maltese}$, for all $0 \leq i \leq k$ and moreover, $\pi_n(\sigma)$ is a $\mathcal{MI}_n$-descriptor. In case (i), we call σ *linear*, and in case (ii) we call it *circular*. We say that σ is *assembled* if $\pi_n(\sigma)$ is an assembled $\mathcal{MI}_n$-descriptor.

Example 6.3. (i) $\sigma_1 = 22M_233M_3\ eI_2\overline{I}_3bM_122$ is an assembled circular Σ_3-descriptor.
(ii) $\sigma_2 = I_022M_233I_1\overline{2}\,\overline{2}\,\overline{M}_1\,\overline{b}I_233\ M_3eI_3$ is a linear micronuclear Σ_3-descriptor.
(iii) $\sigma_3 = 22M_23M_3eI_2\overline{I}_3bM_122$ is not a Σ_3-descriptor.

Every MDS of micronuclear ciliate genes is flanked at its both ends by a pointer or a marker. We denote the pointers flanking MDS M_i by writing $iiM_i(i+1)(i+1)$. For M_1 and M_n we write bM_122 and nnM_ne, respectively. We use a double letter notation for pointers in order to deal with splicing in a simple way in Section 6.2. Formally, to associate a Σ_n-descriptor to a $\mathcal{MI}_n$-descriptor, consider the morphism $\phi_n : (\mathcal{M}_n \cup \mathcal{I}_n)^{\maltese} \to \Sigma_n^{\maltese}$ defined as follows: $\phi_n(I) = I$, for all $I \in \mathcal{I}_n$; $\phi_n(M_i) = iiM_i(i+1)(i+1)$, for all $2 \leq i \leq n-1$; $\phi_n(M_1) = bM_122$ and $f(M_n) = nnM_ne$. For any $\mathcal{MI}_n$-descriptor δ, we say that $\phi_n(\delta)$ is the Σ_n-descriptor *associated* to δ.

We say that $\phi_n(\delta)$ is a *micronuclear* Σ_n-descriptor if δ is a micronuclear $\mathcal{MI}_n$-descriptor. Note that all micronuclear Σ_n-descriptors are linear.

Example 6.4. The micronuclear Σ_9-descriptor of the actin I gene in S.nova, see Example 6.2, is $I_0 33 M_3 44 I_1 44 M_4 55 I_2 66 M_6 77 I_3 55 M_5 66 I_4 77 M_7 88 I_5 99 M_9 e I_6 \overline{33} \overline{M}_2 \overline{2}\,\overline{2} I_7 b M_1 22 I_8 88 M_8 99 I_9$.

For any micronuclear Σ_n-descriptor σ and any $p \in \mathcal{P}_n$, σ contains two occurrences from the set $\{pp, \overline{p}\,\overline{p}\}$: pp represents the pointer in the beginning of MDS M_p and at the end of M_{p-1}, while $\overline{p}\,\overline{p}$ is its inversion.

6.2 Invariants

We give in this section a number of invariants of the gene assembly process: the circularity of the assembled gene (whether or not the gene is assembled on a circular molecule), with the IES-context of the gene (the sequence of IESs preceding and succeeding the assembled gene), but also with the set of molecules excised during assembly. It is worth emphasizing that we establish all these properties based solely on the generic paradigm of pointer-directed assembly, independently of the specificities of either the intra$-$, or the inter$-$molecular model.

During the pointer-directed assembly, ciliates allegedly align their DNA molecules along their pointers, and through recombination they sort the MDSs in the orthodox order. It is essential to observe that in this process, the two strands of any pointer p will be separated: one strand will remain with the block preceding the pointer, while the other strand will remain with the block succeeding the pointer. The single strands will then recombine with the complementary strands obtained by separating in a similar way the second occurrence of p in the gene. This splicing on pointers can be formalized by a word-cutting operation defined in the following.

Let σ be a Σ_n-descriptor. If σ is linear, $\sigma = \alpha_0 p_1 p_1 \alpha_1 p_2 p_2 \alpha_2 \dots \alpha_{k-1} p_k p_k \alpha_k$, with $p_i \in \mathcal{P}_n \cup \overline{\mathcal{P}}_n$, $\alpha_i \in (\Sigma_n \setminus \mathcal{P}_n)^{\maltese}$, then $W_\sigma = \{\alpha_0 p_1, p_k \alpha_k, p_i \alpha_i p_{i+1} \mid 1 \leq i < k\}$. If σ is circular, $\sigma = p_1 \alpha_1 p_2 p_2 \alpha_2 \dots \alpha_{k-1} p_k p_k \alpha_k p_1$, then $W_\sigma = \{p_i \alpha_i p_{i+1}, p_k \alpha_k p_1 \mid 1 \leq i < k\}$. For any set $S \subseteq \Sigma_n^{\maltese}$, we denote $W_S = \cup_{\sigma \in S} W_\sigma$. Note that the set W_σ is equivalent to the set of edges of genome graphs (Section 5.6) and to the reality edges of reduction graphs (Section 5.2).

It is important to note that we do not conjecture that ciliates split their genes by cutting *simultaneously* on each pointer, to yield on the scale of 10^5 MDSs and IESs, followed then by a precise reassembly of all these blocks. Indeed, it is difficult to imagine that such a mechanism would lead to the precise effective assembly that we see in ciliates. Here we merely represent those pointer-delimited coding and non-coding blocks that will be eventually reshuffled to assemble the gene. Our main result states that, given the fixed order in which MDSs must be assembled, the pointer-directed assembly of all the other blocks (IESs) is uniquely determined by the micronuclear structure of the gene.

Example 6.5. For the Σ_9-descriptor σ in Example 6.4, we have

$$W_\sigma = \{I_0 3, 3M_3 4, 4I_1 4, 4M_4 5, 5I_2 6, 6M_6 7, 7I_3 5, 5M_5 6, 6I_4 7, 7M_7 8, 8I_5 9,$$
$$9M_9 eI_6 \overline{3}, \overline{3}\,\overline{M}_2 \overline{2}, \overline{2}I_7 bM_1 2, 2I_8 8, 8M_8 9, 9I_9\}.$$

Our invariant theorem may be stated now as follows.

Theorem 6.6 ([67]). *Let σ be a micronuclear Σ_n-descriptor.*

(i) There exists a set $\mathcal{A}_\sigma$ of Σ_n-descriptors such that
  (a) $W_\sigma \cup \overline{W_\sigma} = W_{\mathcal{A}_\sigma} \cup \overline{W_{\mathcal{A}_\sigma}}$;
  (b) there exists an assembled Σ_n-descriptor in $\mathcal{A}_\sigma$;
(ii) For any other set S of Σ_n-descriptors, if S satisfies conditions (a)-(b) above, then $S \approx \mathcal{A}_\sigma$.

Moreover, $\mathcal{A}_\sigma$ consists of exactly one linear Σ_n-descriptor and possibly several circular ones.

Theorem 6.6 may be stated informally by saying that the final result of gene assembly, including the molecule where the assembled gene is placed, as well as all the other non-coding molecules excised in the process, is unique.

Example 6.7. Consider the Σ_9-descriptor σ associated to gene actin I in S.nova in Example 6.4, with W_σ given in Example 6.5. It follows from Theorem 6.6 that the results of assembling the gene are

$$\{I_0 33\overline{I}_6 \overline{e}\overline{M}_9 \overline{99M}_8 \overline{88M}_7 \overline{77M}_6 \overline{66M}_5 \overline{55M}_4 \overline{44M}_3 \overline{33M}_2 \overline{22M}_1 \overline{b}\overline{I}_7 22I_8 88I_5 99I_9,$$
$$4I_1 4\}.$$

Thus, the non-coding block $4I_1 4$ is excised as a circular molecule and the gene is assembled linearly in the inverse order from $\overline{M}_9$ to $\overline{M}_1$ with the non-coding block $I_0 33\overline{I}_6$ preceding it and $\overline{I}_7 22I_8 88I_5 99I_9$ succeeding it.

Note that Theorem 6.6 holds both for the intra-, and the inter-, molecular models for gene assembly. Consequently, the set of molecules generated by the assembly cannot be used to distinguish between different assembly strategies, either intramolecular, or inter-molecular. Instead, to (in)validate either model, one could experimentally identify the sets of molecules generated at various stages of the assembly and verify it against the predictions made by the two models.

Our results hold also in a more general way. We have proved that the final result $\mathcal{A}_\sigma$ of assembling a micronuclear Σ_n-descriptor σ is unique modulo conjugation and inversion. As a matter of fact, our proofs apply unchanged also to the following variant proved in [10] for the intramolecular model. Let $P \subseteq \mathcal{P}_n$. There exists a unique (modulo conjugation and inversion) set $\mathcal{A}_{P,\sigma}$ of Σ_n-descriptors with the following property: $M_{p-1}ppM_p \leq \alpha$ for some $\alpha \in \mathcal{A}_{P,\sigma} \cup \overline{\mathcal{A}_{P,\sigma}}$ if and only if $p \in P$. In other words, if the assembly is to be done only on a given set P (that may be different from the total set $\mathcal{P}_n$), then the result is unique. To prove the result, it is enough to replace the morphism ϕ_n in Section 6.1 with a morphism $\phi_{P,n}$ that only inserts pp in case $p \in P$. This extension of Theorem 6.6 does not contradict the non-determinism of gene assembly: the ciliate may choose to reduce the pointers in any order. The result above only says that after assembling on a *fixed* set of pointers, the result, including the excised molecules, is unique.

7 Template-guided recombination

In the previous sections we have considered models which take an abstract view of the gene assembly process in ciliates. We now move to a lower, implementation-oriented, level of abstraction in which we attempt to begin addressing the question of *how* the assembly process takes place in vivo. Our quest for the discovery of the "biological hardware" responsible for implementing assembly begins with a simple examination of how MDSs might overlap to fit together, in the correct order, while also removing IESs. As has been noted above, each MDS is flanked by pointer sequences – that is, looking at the level of DNA sequence, there will be a proper suffix of MDS n which is equal to a proper prefix of MDS $n+1$. Considering this type of structure for an entire gene of several MDSs, a computer scientist will immediately recognize the *linked list data structure*: the core of the MDS is the data while the pointer sequence indicates the "address" of the next data item (MDS). The initial state of MDSs, distributed throughout the MIC, is reminiscent of the non-linear distribution of linked list data in heap memory. It is worth noting that the ciliate data structure is, in fact, significantly more sophisticated than a classic linked list; whereas each element of a linked list contains two separate components, an area for a data payload and an area for a pointer to the next element, the ciliate version of the linked list actually combines these two elements. Since the pointers always lie within the MDSs, the pointer to the next MDS is also part of the "data" contained in the MDS and is fully integrated into the assembled gene.

Completing the process of gene assembly, starting with MIC and ending with MAC, can be seen as implementing a linked-list specification. However, it does not appear that the pointers alone are guiding this implementation process as some of them may be too short to serve as a unique pointers to the following MDSs. Yet the pointer sequences are still always present. A natural question to be answered by any suggested implementation of gene assembly is thus: "what role do the pointers play and why are they present?".

A realistic, biologically implementable, model must incorporate a number of principal features. It must be *irreversible*, it must be *self-propagating* or *reusable*, it cannot be sequence specific, i.e., it cannot rely on the presence of certain fixed sequences, as a huge variety of pointer sequences are known (instead it must be *configuration specific*), and it must have some mechanism for *identifying the MDS/IES boundary and hence the pointers*. The basic DNA-template model of *template-guided recombination* was introduced in [76] to address exactly these requirements which we will clarify in more detail after describing the model.

7.1 DNA template-guided recombination

We consider here a schematic view of DNA as a picket fence with the sugar-phosphate backbones running horizontally along the top and bottom of the strand and the hydrogen bonds running vertically between the backbones. Suppose now that we wish to assemble two strands of DNA, X and Y, having the sequences $X_1\alpha\beta\delta X_2$ and $Y_1\varepsilon\beta\gamma Y_2$, respectively, where $\beta = \beta_1\beta_2$. In order to guide this assembly, we

assume the existence of a DNA template T of the form $T_1\alpha\beta\gamma T_2$ which is placed in-between the two target strands X and Y, as seen in Fig. 21. Note that we use the notation $\bar{\alpha}$ to denote the Watson-Crick complement of the DNA sequence α and that, in Fig. 21, the $\alpha\beta$ region of X is now aligned with $\overline{\alpha\beta}$ on the template; similarly, $\beta\gamma$ of Y is aligned with its complement on the template. At the same time, the sequence of δ, beginning from the first nucleotide, must not be complementary to γ and, in a similar way, ϵ must not be complementary to α. It is important to note that X and Y need not be physically disconnected independent strands but may instead be different regions of a single, connected, strand of DNA.

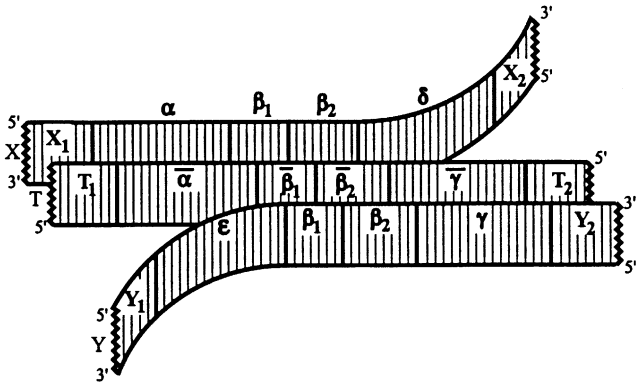


Fig. 21. Sequences $X_1\alpha\beta\delta X_2$ and $Y_1\varepsilon\beta\gamma Y_2$ stacked with DNA template $T_1\alpha\beta\gamma T_2$ in-between. [76]

The template-guided recombination now takes place in these principle steps:

- The hydrogen bonds in the $\alpha\beta_1$ region of X and the $\overline{\alpha\beta_1}$ in the template are broken and switch from binding the backbones within X and Y vertically to binding complementary sequence horizontally between X and Y; likewise for the second half of the template. In our fence analogy, the vertical pickets within DNA double-strands are replaced by floors and roofs *across* DNA strands, as pictured in Fig. 22.
- Cuts are now made in the backbones of the roof/floor assembly (at the ends of the roof/floor structures) to yield the free-standing structure of Fig. 23.
- The same cuts also yield a new copy of the original template strand, shown in Fig. 24, and the strands $Y_1\varepsilon\beta_1$ and $\beta_2\delta X_2$ (not pictured) which are left free to float away.
- The roof and floor structures of Fig. 23 rotate to align the cut backbones which are then healed (via ligation) yielding the complete double stranded DNA $X_1\alpha\beta\gamma Y_2$ which is the recombination of the prefix $X_1\alpha\beta_1$ of X and the suffix $\beta_2\gamma Y_2$ of Y – thus X and Y have been recombined.

- Likewise, the roof and floor structures of Fig. 24 rotate, align and ligate to yield $T_1\bar{\alpha}\bar{\beta}\bar{\gamma}T_2$ – thus the template is reconstituted.

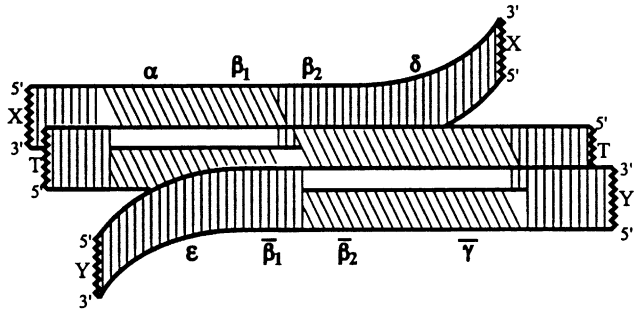


Fig. 22. Hydrogen bonds switch from being vertical pickets holding individual strands together to forming floors and roofs across strands. [76]

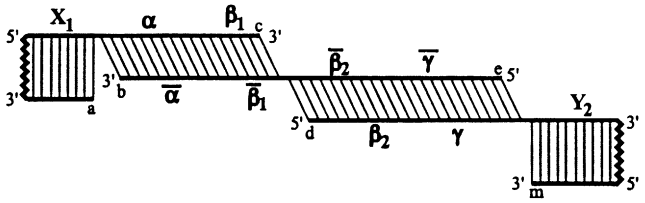


Fig. 23. One of the products resulting from cuts made in the backbones of the configuration of Fig. 22. [76]

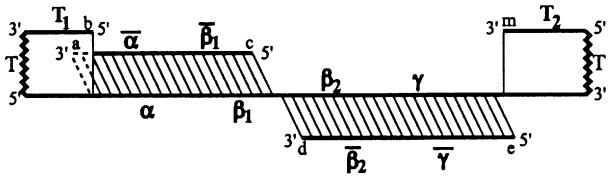


Fig. 24. The reconstituted template, resulting from cuts made in the backbones of the configuration of Fig. 22. [76]

It can be seen clearly that the model, when considered in symbolic terms (as often done in computer science), meets our requirements for a biologically implementable system. The process of template-guided recombination may be described by the following implication:

$$\alpha\beta\delta + \alpha\beta\gamma + \varepsilon\beta\gamma \Rightarrow \alpha\beta\gamma + \alpha\beta\gamma + \varepsilon\beta_1 + \beta_2\delta$$

where $\beta_1\beta_2 = \beta$. On the left side we have the first term consisting of the required subsequences for X, the second term is the template and the third term is the required subsequence of Y. On the right side, the first term expresses the form of X recombined with Y, the form of the reconstituted template and the last two terms are the forms of the "loose ends". Note that we must have the components on the left side present in order for the reaction to take place. If they are present, this equation describes *what* happens, though not how it happens, and what we obtain is the four products on the right side. Note carefully that from this moment on, no three components of the right hand side have the form required of the three components of the left hand side – thus the process is irreversible (one-way).

Examining the right side, we see $\alpha\beta\gamma$ twice, demonstrating that the template in this model is self-propagating: we begin with one component $\alpha\beta\gamma$ on the left hand side and after a single iteration we get two such components. Hence as the process progresses iteratively, we have an explanation of the growth of the number of available templates. Thus even if we begin with only a single copy of the template, we very quickly end up with an "abundance" of templates. This is necessary as we recall from the section on ciliate biology that many copies of the MIC chromosomes are present during the polytene chromosome stage so that multiple copies of each template are required to successfully assemble a full MAC genome.

Further, it is clear that the whole three step process does not depend on *specific* sequences α, β, γ; indeed, all that matters is the relationship between the sequences that causes the formation of a particular configuration.

Considering this process carefully, the true nature of pointers becomes apparent: pointers are sequence segments within which the transfer of roofs, and dually the transfer of floors, takes place. Consequently, the most essential backbone cuts of the recombination process will take place in the pointer region. Pointers are records of transfers. Hence, in our scheme, pointers are the regions denoted $\beta = \beta_1\beta_2$ while $\alpha\beta$ and $\beta\gamma$ correspond to MDSs M_i and M_{i+1}.

7.2 RNA template-guided recombination

A variant of DNA template-guided recombination has been proposed in [5] which considers *RNA templates*. Following through the steps of the DNA template model one can see that a portion of the template strand ends up being incorporated into the final assembled product; this presents no problem for DNA templates but is infeasible with RNA templates since DNA and RNA backbones are incompatible. Rather than proposing a template which sits "in-between" the strands to be assembled, the RNA template model suggests a template which "hangs above" the strands to be recombined, guiding the recombination but never directly participating in it.

Consider DNA strands $\alpha\beta\delta$ and $\varepsilon\beta\gamma$ again containing the MDSs $\alpha\beta$ and $\beta\gamma$ and a double-stranded RNA template molecule $\alpha\beta\gamma$. We stack the two substrate strands in tandem, as in the DNA model, but place the template horizontally above, rather than between, the substrates as in Fig. 25. The $\bar{\beta}$ sequence of the RNA template begins to form hydrogen bonds with the sequence β in one of the substrates; note that while the backbones are incompatible, it is certainly possible for RNA and DNA to share hydrogen bonds across two strands. The complementary strand of the template RNA similarly binds to the other substrate strand. With the internal "picket fence" hydrogen bonds of the substrate strands broken, the complementary portions at the bottom of each strand now form a floor of hydrogen bonds. If the RNA template is now removed, the complementary strands at the top of the substrates will form a roof of hydrogen bonds. Cutting the DNA backbones in the four places indicated in Fig. 25 and rotating, followed by healing (ligation), the broken backbones similarly to the DNA model yields a correctly assembled DNA strand.

Note carefully that the RNA template is not integrated into the resulting structure; rather, the RNA template served only to break up the hydrogen bonds in the β region of the substrate strands, inducing the formation of a floor of hydrogen bonds between the two substrates which then induced the formation of a complementary roof structure following the removal of the RNA template.

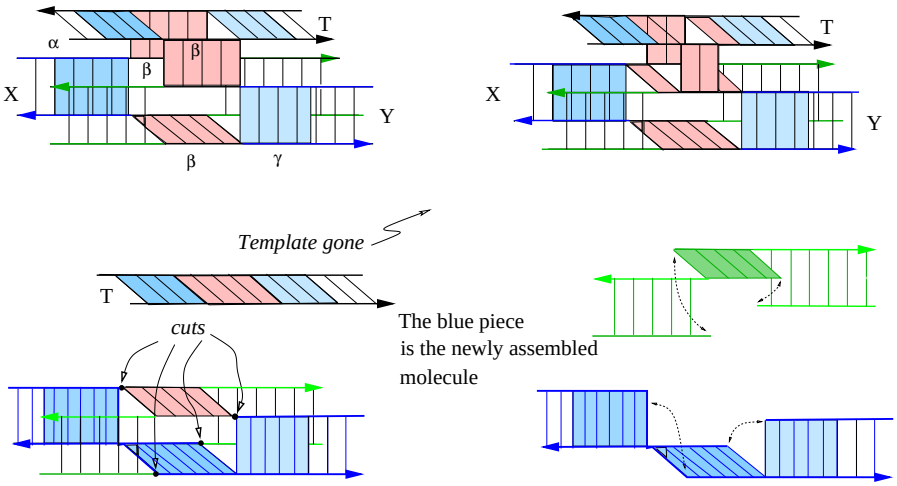


Fig. 25. Reading left to right, top to bottom: The RNA template forms hydrogen bonds with β and $\bar{\beta}$ in the substrate strands, causing the formation of a floor between strands. When the RNA template is removed, a roof structure is now formed. Four cuts are made, followed by backbones swinging back into position and being healed to form the assembled strand. [5]

Yet another similar approach requires only a single-stranded RNA template, which is placed "diagonally" between the two DNA picket fence strands (details are given in [5]).

The development of theoretical models of template-guided recombination has helped to direct biological inquiry into core questions surrounding gene assembly. It is natural, from a biological perspective, to ask where templates might originate. Both the RNA and DNA template-guided models suggest that templates are composed of sequence which is highly similar to that in the old MAC. Recent experimental results in the ciliate *Oxytricha trifallax* (*Sterkiella histriomuscorum*), guided by the insight provided in the theoretical models, support the hypothesis that short MAC-specific RNA templates are involved in gene assembly [66]. We point out also that the appendix of [5] contains a straight-forward RNA-template-combination explanation for the molecular operations ld, hi, and dlad introduced in Section 5 which are the basis of the intramolecular models presented in this chapter.

7.3 Template Guided Recombination on Words and Languages

We move now to consider theoretical research on the template model. Together with combinatorial models discussed in preceding sections, the theoretical work concerning the template model provided both some new insights into the nature of gene assembly and a whole spectrum of novel and interesting notions, models, and results for theoretical computer science.

A formal language theoretic version of the basic *DNA template-guided recombination* (abbreviated TGR) operation of [76] was first studied in [20]. For words $x, y, z, t \in \Sigma^*$ and natural numbers $n_1, n_2 \geq 1$, we denote by z, the product of the recombination of x and y, guided by template t, by $(x, y) \vdash_{t,n_1,n_2} z$. More specifically, if $x = u_1\alpha\beta v_1, y = v_2\beta\gamma u_2, t = \alpha\beta\gamma$ with $\alpha, \beta, \gamma, u_1, u_2, v_1, v_2 \in \Sigma^*$, $|\alpha|, |\gamma| \geq n_1$ and $|\beta| = n_2$, then we may write the TGR product $z = u_1\alpha\beta\gamma u_2$. If $T, L \subseteq \Sigma^*$ are languages, then $\pitchfork_{T,n_1,n_2}(L)$ is defined by

$$\pitchfork_{T,n_1,n_2}(L) = \{z \ : \ \exists x, y \in L, t \in T \text{ such that } (x, y) \vdash_{t,n_1,n_2} z\}.$$

The shorthand notation $\pitchfork_T(L)$ is used whenever n_1, n_2 are understood. We note that the restriction on pointer length, $|\beta| = n_2$, is not as strict as it appears since it has been proven equivalent to the restriction $|\beta| \geq n_2$ in [20].

Given the nature of biochemical reactions, it is natural to consider an iterated version of TGR as well: let $\pitchfork^0_{T,n_1,n_2}(L) = L$ and for all $i \geq 1$, let

$$\pitchfork^i_{T,n_1,n_2}(L) = \pitchfork^{i-1}_{T,n_1,n_2}(L) \cup \pitchfork_{T,n_1,n_2}(\pitchfork^{i-1}_{T,n_1,n_2}(L)).$$

Then we also define $\pitchfork^*_{T,n_1,n_2}(L)$ as

$$\pitchfork^*_{T,n_1,n_2}(L) = \bigcup_{i \geq 0} \pitchfork^i_{T,n_1,n_2}(L).$$

Finally, let $\mathfrak{L}, \mathfrak{T}$ be classes of languages and $n_1, n_2 \geq 1$. We define the following closure classes:

$$\pitchfork_{\mathfrak{T},n_1,n_2}(\mathfrak{L}) = \{\pitchfork_{T,n_1,n_2}(L) \ : \ T \in \mathfrak{T}, L \in \mathfrak{L}\},$$
$$\pitchfork^*_{\mathfrak{T},n_1,n_2}(\mathfrak{L}) = \{\pitchfork^*_{T,n_1,n_2}(L) \ : \ T \in \mathfrak{T}, L \in \mathfrak{L}\}.$$

We denote the families of finite languages by FIN and regular languages by REG, and recall that a family of languages is said to be a full AFL if it is closed under homomorphism, inverse homomorphism, intersection with regular languages, union, concatenation and Kleene plus.

On initial inspection of TGR, there appears to be a similarity with the well-known model of splicing systems, but this relationship has been shown to be superficial in [20]. Before we state the relevant formal result, let us recall the basic operational scheme of splicing systems.

A *splicing* scheme or *H scheme* is a pair $\sigma = (\Sigma, R)$ where Σ is an alphabet and $R \subseteq \Sigma^*\#\Sigma^*\$\Sigma^*\#\Sigma^*$ is a set of splicing rules where $\$, \#$ are not elements of Σ. For a rule $r \in R$, we define the relation $(x, y) \models_r z$ if $r = u_1\#u_2\$u_3\#u_4, x = x_1u_1u_2x_2, y = y_1u_3u_4y_2, z = x_1u_1u_4y_2$, for some $u_1, u_2, u_3, u_4, y_1, y_2, x_1, x_2 \in \Sigma^*$.

For a language $L \subseteq \Sigma^*$ and an H scheme $\sigma = (\Sigma, R)$, we define $\sigma(L) = \{z \in \Sigma^* \ : \ \exists x, y \in L, r \in R$ such that $(x, y) \models_r z\}$ and extend to iterated splicing as follows: let $\sigma^0(L) = L$ and $\sigma^i(L)$ be defined by $\sigma^i(L) = \sigma^{i-1}(L) \cup \sigma(\sigma^{i-1}(L))$ for all $i \geq 1$. Finally, as expected,

$$\sigma^*(L) = \bigcup_{i \geq 0} \sigma^i(L).$$

For classes of languages $\mathfrak{L}, \mathfrak{R}$, let $H(\mathfrak{L}, \mathfrak{R}) = \{\sigma^*(L) \ : \ L \in \mathfrak{L}, \sigma = (\Sigma, R), R \in \mathfrak{R}\}$.

Lemma 7.1 ([19]). *For all $n_1, n_2 \geq 1$, for all full AFLs $\mathfrak{L}$:*

$$\mathfrak{L} = \Cap^*_{\text{FIN},n_1,n_2}(\mathfrak{L}) = H(\mathfrak{L}, \text{FIN}),$$

while for all finite languages and templates:

$$\text{FIN} \subset \Cap^*_{\text{FIN},n_1,n_2}(\text{FIN}) \subset H(\text{FIN}, \text{FIN}) \subset \text{REG}.$$

Despite this result, it has been shown in [20] that every regular language is the coding of a language in $\Cap^*_{\text{FIN}}(\text{FIN})$ demonstrating a relatively modest computational power for single-application TGR.

To investigate the computational power of the iterated case, it is necessary to define the notion of a "useful" template; we say that a template $t \in T$ is *useful* on L, n_1, n_2 if there exists $u_1\alpha\beta v_1, v_2\beta\gamma u_2 \in \Cap^*_{T,n_1,n_2}(L)$ with $|\alpha|, |\gamma| \geq n_1$, $|\beta| = n_2$, $u_1, u_2, v_1, v_2 \in \Sigma^*$ and $t = \alpha\beta\gamma$. If every template $t \in T$ is useful on L, n_1, n_2, then we say that T is useful on L, n_1, n_2. The following results, demonstrating the surprisingly limited power of iterated TGR, were shown in [19]:

Theorem 7.2 ([19]). *Let $n_1, n_2 \geq 1$, $\mathfrak{L}$ be a full AFL and $L, T \in \mathfrak{L}$. If T is useful on L, n_1, n_2, then $\Cap^*_{T,n_1,n_2}(L) \in \mathfrak{L}$.*

Corollary 7.3 ([19]). *Let $n_1, n_2 \geq 1$. For all full AFLs $\mathfrak{L}$, $\Cap^*_{\text{REG},n_1,n_2}(\mathfrak{L}) = \mathfrak{L}$.*

The problem of template equivalence, viz. "Given sets of templates T_1, T_2, are $\pitchfork_{T_1}$ and $\pitchfork_{T_2}$ identical operations?" has been considered in [27] which gives a characterization of when two sets of templates define the same TGR operation in formal language theoretic terms, leading the following decidability result:

Theorem 7.4 ([27]). *Let $n_1, n_2 \geq 1$ and $T_1, T_2 \subseteq \Sigma^*$ ($|\Sigma| \geq 3$) be regular sets of templates. Then it is decidable whether or not $\pitchfork_{T_1,n_1,n_2}(L) = \pitchfork_{T_2,n_1,n_2}(L)$ for all $L \subseteq \Sigma^*$.*

Several variants of TGR have been studied as well, including a computationally universal version with added deletion contexts [21] and a purely intramolecular version which resembles a templated version of the ld operation [15] discussed above.

In addition to the very literal formalization of TGR considered in this section, the underlying theoretical model has also inspired work at a more abstract level.

7.4 Covers from templates

The process of gene descrambling may be abstractly formulated in simple terms as a procedure which takes MDSs from the MIC and connects them, via overlap, to form the MAC. The template-guided recombination model discussed above provides a concrete suggestion of how this process might be implemented; we view a template, T, as a sort of magnet which glues together regions of MIC chromosomes containing MDSs to form orthodox MAC genes. Returning to a more abstract level, we now consider the set of all MDSs as our primary object of study. In this view, a template is now a request: "with this set of segments(MDSs), please cover me": and therefore our core question is now "given a set of segments, how can a particular word be covered with these segments?". This leads naturally to the study of various properties of coverings, and the notion of uniqueness of coverings formalized as *scaffolds* presented in [35].

An *interval* is a set of integers of the form $\{n, n+1, \ldots, n+m\}$, where $n \in \mathbb{Z}$, and $m \in \mathbb{N}$. For a given alphabet Σ, we define a *segment* as a function $f : A \to \Sigma$, where A is an interval, and denote the set of all segments over Σ by $\mathcal{S}_\Sigma$. Since f is a function, we alternatively view segments as sets of ordered pairs of the form $(n, f(n))$, called *elements of* f, with n called the *location* of $(n, f(n))$; thus elements of f are ordered through their locations. This point of view is very convenient as it provides a set-theoretical calculus of segments: we can consider inclusions, union, intersections, differences, ... of segments. Also, in this way, a set of segments is a family of sets.

For a set $C \subseteq \mathcal{S}_\Sigma$ and a segment $f \in \mathcal{S}_\Sigma$, we say that C *covers* f if $f = \bigcup C$. Intuitively, f is covered by C if each element of f is present in at least one segment of C, and all elements of all segments of C are present in f. Note that in general an element of f may be present in several segments of C, i.e., the segments of C may overlap. One may also have redundant segments in C, i.e., segments which cover only elements of f that are already covered by other segments of C. We thus say that a cover C of f is *tight* if for every $z \in C$, $C - \{z\}$ is not a cover of z.

Often covers are chosen from a subset of $\mathcal{S}_\Sigma$. Given such an $F \subseteq \mathcal{S}_\Sigma$, and a cover C of f with $C \subseteq F$, we say that C is a *small cover* of f (w.r.t F) if $|C| \leq |Z|$ for all $Z \subseteq F$ covering f. Note that the property of being a small cover is a global property: C is a small cover of f w.r.t. F if any other cover Z of f, $Z \subseteq F$, has at least as many segments as C. The set of all small covers of f w.r.t. F is denoted $SC_F(f)$, and the *small index* of f (w.r.t. F) is the cardinality of the small covers of f (w.r.t. F). For any small cover C of f w.r.t. F, we get a natural order $C(1), \ldots, C(m)$ of C, where m is the small index of f, and the order is determined by increasing locations of first elements of the segments of C.

Example 7.5. Let $\Sigma = \{a, b, c\}$ and $f \in \mathcal{S}_\Sigma$ be defined by

$$f = \{(3, a), (4, b), (5, b), (6, a), (7, c)\}$$

which may be abbreviated as $f = (3, abbac)$ since f begins at location 3.

Consider the sets $F = \{(3, ab), (4, bb), (4, bba), (5, bac), (6, ac)\}$ and $C = \{(3, ab), (4, bb), (6, ac)\}$. It is clear that C covers f and is tight, since no element of C can be removed while still covering f; however, with respect to F, C is not small since $|C| = 3$ and the set $\{(3, ab), (5, bac)\} \subseteq F$ also covers f and has cardinality 2.

From the point of view of the original biological motivation, the segment f represents a descrambled MAC gene while the set F is the collection of MIC gene fragments available for assembly. We now proceed to investigate the structure of f by considering the family of all small covers of f with respect to some fixed F.

Let $f \in \mathcal{S}_\Sigma$, $F \subseteq \mathcal{S}_\Sigma$ be such that it contains a cover of f, and let m be the small index of f with respect to F. Let $1 \leq i \leq m$. The i-th kernel of f with respect to F (denoted $ker_{i,F}(f)$) is defined by

$$ker_{i,F}(f) = \left(\bigcap_{C \in SC_F(f)} C(i) \right) - \bigcup_{\substack{C \in SC_F(f) \\ j \neq i}} C(j) \ .$$

We now define the *scaffold* of f (with respect to F) as the set $\{ker_{1,F}(f), \ldots, ker_{m,F}(f)\}$ of all kernels of f.

Theorem 7.6 ([35]). *For each $1 \leq i \leq m$, $ker_{i,F}(f)$ is a nonempty segment.*

For any segment f, the choice of F determines a certain natural class of "maximal" subsegments of f; namely, those subsegments which are not strict subsegments of other subsegments. Let $P_F(f)$ be the set of all subsegments of f that belong to F. We say that a segment $g \in P_F(f)$ is *long* (with respect to F) if it is not properly included in any other segment in $P_F(f)$. The set of all long segments of f (with respect to F) is denoted $LP_F(f)$. Additionally, we call a cover long if it consists solely of long segments.

We now turn our attention to the study of the canonical class of covers which have both the "long" and "small" property. For each $1 \leq i \leq m$, let $LP_F(f, i) =$

$\{y \in LP_F(f) \ : \ ker_{i,F} \subseteq y\}$. That is, we categorize the long segments of f with respect to F according to containment of kernels. Since $LP_F(f,i)$ is an ordered set, we let $rt_F(f,i)$ (resp., $lt_F(f,i)$) be the maximal, or rightmost (resp., minimal, or leftmost) element of $LP_F(f,i)$. The following result provides a method for constructing "canonical" small covers of f.

Theorem 7.7 ([35]). *The sets* $\{rt_F(f,1),\ldots,rt_F(f,m)\}$ *and* $\{lt_F(f,1),\ldots,lt_F(f,m)\}$ *are long small covers of* f *with respect to* F.

More detailed analysis of the structure of scaffolds is given in [35].

7.5 Topology based models

An important question that arises from considering template-guided recombination concerns the three dimensional structure of DNA undergoing multiple recombination events. Recently, two new approaches to this question have been undertaken. The physical structure of the DNA strand undergoing recombination is directly considered in [5] through the use of virtual knot diagrams. Micronuclear genes are represented in a schematic form which explicitly denotes only the relative locations of the pointer sequences. Consider, e.g., the Uroleptus gene *USGI* [13] which has the following MDS descriptor: $M_1M_3M_4M_5M_7M_{10}M_{11}M_6M_8M_2M_9$. The corresponding legal string is the following: 2 3 4 4 5 5 6 7 8 10 11 11 6 7 8 9 2 3 9 10. It is now possible to interpret this sequence as a Gauss code. Each symbol in a Gauss code must occur exactly twice and to each code we associate a virtual knot diagram in a similar manner to the construction of the recombination graphs exposited above:

- For each symbol occurring in the code, we place a (disconnected) crossing in the plane and label the crossing with the corresponding symbol. A crossing may be thought of as similar to a vertex with predetermined order of the incident edges of degree 4 in a graph.
- We choose an arbitrary point in the plane to denote as the base point.
- Following a chosen direction, we connect crossings according to the order in the Gauss code. In our example, we would draw arcs from the base point to crossing 2, crossing 2 to crossing 3, crossing 3 to crossing 4, crossing 4 to itself, crossing 4 to crossing 5, and so forth. Each crossing corresponds to a "roof-floor" structure depicted in Fig. 26 left.
- We connect the remaining arc leaving the final crossing back to the base point.

Note that during the construction of the virtual knot diagram it might happen that we have to cross an already sketched arc. This crossing is not labeled and does not correspond to a required pointer-guided homologous recombination, therefore it is called a "virtual crossing". The virtual crossings correspond to a cross-over embedding of the DNA in space when one helix crosses over another.

The virtual knot diagram is now relabeled with each crossing receiving the label of its associated pointer and indicating inverted pointers with a bar. The process of assembly is now reduced to one of *smoothing* the crossings of the virtual knot

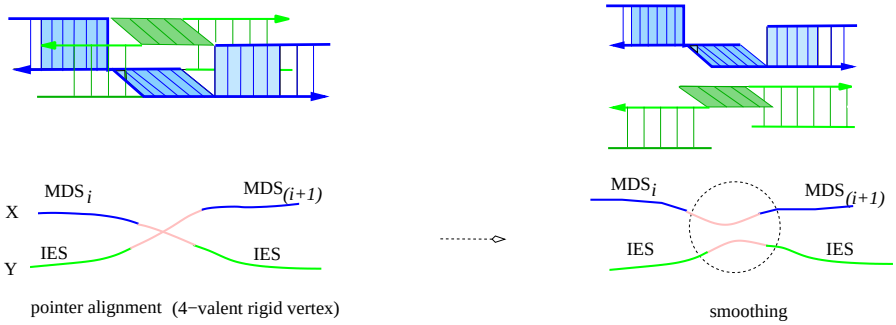


Fig. 26. Roof-floor structure of a crossing in a virtual knot diagram. [5]

diagram (see Fig. 26 right). The smoothing of a crossing consists of eliminating the crossing by splicing together the arcs of the crossing according to the pointers: if the pointers are not inverted, we splice together the arcs following the orientation; if the pointers are inverted, we splice together the arcs opposite to the orientation. The result of a simultaneous smoothing of all crossings is a virtual link diagram with no real crossings remaining; note that the link may be composed of multiple components. If the arcs of the virtual knot diagram are labeled with the respective MDS and IES names (see [5]), then we have the following theorem.

Theorem 7.8 ([5]). *For every labeled virtual knot diagram derived from a representation of a scrambled gene G, there exists a simultaneous smoothing yielding a link with a component C containing a subarc labeled with all MDSs in orthodox macronuclear order.*

Note that the basic mechanism of gene assembly through the smoothing of virtual knot diagrams was already introduced in [31] in terms of a graph-based model, which was also briefly discussed in Section 5.6. This model is based on the notion of recombination graph representing pointers as vertices, and MDSs and IESs as edges between the vertices standing for their flanking pointers. The recombination graph is first subject to a process of *graph folding* yielding a structure similar to the virtual knot diagram. A subsequent step of *graph unfolding* is similar to that of *smoothing* above and yields a representation of the assembled gene and of all the molecules excised during gene assembly. In this way, the approach of virtual knot diagrams from [5] is a translation of the graph-theoretic approach from [31] into a topological framework. This allows one to apply a rich set of techniques from both graph theory and knot theory to the investigation of gene assembly.

One significant issue which has not yet been addressed concerns the thermodynamics of template-guided recombination. For template-guided recombination to be implemented as suggested in the theoretical models requires some intricate positioning and biochemical operations; specifically, it requires the juxtaposition of three nucleic acid strands in space followed by a strand branch migration process as is observed in vivo (see, e.g., [26, 83]) and in vitro (see, e.g., [81, 82]). In order for such

branch migration process to start, it is possibly necessary that some of the cuts in the molecule (Fig. 25 bottom left) appear early in the process. On the other side, it is possible to suppose that the juxtaposition of the molecules is artificially created by as-yet undiscovered enzymatic mechanics. Yet another possibility would be a simple argument indicating the thermodynamic favorability of the process. In the absence of such complex additional enzymatic machinery, template-guided recombination must rely upon diffusion processes within the cell to juxtapose the template and substrate strands. It is a well known theorem in mathematics that random walks in three-space need never pass through the same point twice, so the likelihood of juxtaposing three molecules, in three-space, seems low. An alternative knot-theory based model presented in [22] attempts to reduce the complexity of this problem by considering the effects of individual recombinations on the structure of the substrate DNA.

When circular DNA undergoes recombination, it is supercoiled due to twist applied by the recombination machinery [57, 80]. Where linear DNA has unbound ends which are free to rotate and relax, thus removing the induced twist, circular DNA cannot relax in this fashion after recombination. Instead, the relaxation of the DNA strand induces supercoiling of the DNA molecule. The connection between linking, twist, and writhe of the DNA has been observed decades ago [80], therefore, multiple recombinations on a closed circle of DNA can lead to a knotted, supercoiled, strand.

It is suggested in [22] that DNA is assembled in a closed circular topology where the twist induced by a template-guided recombination (or multiple recombinations) causes a new, potentially knotted and supercoiled, topology to form. This new topology could facilitate the juxtaposition of the "next" regions of DNA to be assembled.

8 Discussion

In this chapter we have discussed a number of topics related to research on the computational nature of gene assembly in ciliates, including the two main models for gene assembly. Among others, we have discussed their mathematical formalizations, invariant properties, template-based DNA recombination, and topology-based models for gene assembly. Due to space restrictions, we could not discuss all lines of research; for the sake of completeness, we now mention briefly some of the topics that were not covered.

The organization of the micronuclear genes into broken and shuffled MDSs, separated by IESs is one of the characteristic features of the ciliates. To explain the evolutionary origin of this organization, a somewhat geometrical hypothesis based on a novel proposal for DNA repair is suggested in [34].

Approaches based on formal languages have been introduced for both the intermolecular model and the intramolecular model, leading to very diverse research topics: computability, language equations, closure properties, hierarchies of classes of languages, etc. A typical approach is to consider contextually-based applications of string rewriting rules, see, e.g., [58] for the intermolecular model. A derivation relation and an axiom are introduced, thus obtaining an acceptance mechanism: start with a multiset of strings, e.g., consisting of several copies of the input string and

eventually derive a multiset containing the axiom. It can be proved that such a mechanism is a universal computing device. A similar result can be obtained also based on the intramolecular model, see [51], where the multisets are replaced with single strings. The idea is to concatenate several copies of the input string rather than having them in a multiset. A generating, rather than accepting, computing device inspired by gene assembly was also considered in [25]. Defining non-contextual string-based rewriting rules inspired by the molecular operations in either model leads to language operations and to questions related to closure properties, or solutions of language equations, see [16], [23], and [36], and also [24] for a more general framework. Language operations inspired by the template-based DNA recombination were considered in [20] and in [27].

The original, non-contextual, intra- and inter-molecular operations were generalized to the synchronized insertion and deletion operations on linear strings in [16]. Let α, β be two nonempty words in an alphabet Σ^*. The synchronized insertion of β into α is defined as: $\alpha \oplus \beta = \{uxvxw \mid \alpha = uxw, \beta = vx, x \in \Sigma^+, u, v, w \in \Sigma^*\}$. while the synchronized deletion of β from α is defined as: $\alpha \ominus \beta = \{uxw \mid \alpha = uxvxw, \beta = vx, x \in \Sigma^+, u, v, w \in \Sigma^*\}$. All language families in the Chomsky hierarchy were shown to be closed under synchronized insertion while only the families of regular and recursively enumerable languages were closed under synchronized deletion. The existence of a solution was shown to be decidable for language equations of the form $L \odot Y = R$ and $X \odot L = R$ where $\odot$ is one of synchronized insertion or synchronized deletion operations and L, R are regular languages. The same problems are undecidable in the case that L is a context-free language.

More general results considering families of languages defined by reversal-bounded counter machines are also given in [16] along with results for a similarly generalized version of the hi operation. Generalized versions of the ld and dlad operations are considered in [17] while families of languages defined by closure under these generalized operations are examined in [18].

Two novel classes of codes based on synchronized insertion were defined and studied in [14]. The synchronized outfix codes ($\oplus$-codes) defined by the equation $(L \oplus \Sigma^+) \cap L = \varnothing$ and the synchronized hypercodes ($\otimes$-codes) defined by $(L \otimes \Sigma^+) \cap L = \varnothing$ (where $\otimes$ is the transitive closure of $\oplus$, namely synchronized scattered insertion). The $\oplus$-codes and $\otimes$-codes are shown to be completely disjoint from the regularly studied classes of $*$-codes and it is demonstrated that it is decidable if a regular language is an $\oplus$-code while the same property is undecidable for linear context-free languages. It is, however, decidable if an arbitrary context-free language is an $\otimes$-code while this same property is, unsurprisingly, undecidable for context-sensitive languages.

A different computability approach was developed in [3], where the process of gene assembly is used to solve an NP-complete problem. Conceptually, this is important since gene assembly is confluent, see Section 5, i.e., it yields a computing device that answers 'yes' to all legal inputs. A way around the problem is to make the device highly non-deterministic by extending its set of legal inputs. For example, strings with more than two occurrences of each letter may be allowed. With this modification, a suitably defined model can be introduced to solve the Hamiltonian path

problem (HPP) by mimicking gene assembly on an encoding of the input to HPP, see [3]. The intermolecular model also leads to solutions to computational problems, see [50] for a solution to the satisfiability problem. Yet another approach based on boolean circuits was investigated in [52]. Algorithmic questions related to finding a gene assembly strategy were considered in [49].

The topological model of gene assembly with virtual knot diagrams raises new mathematical questions. Considering that a virtual knot diagram could represent the physical structure of the micronuclear DNA at the time of recombination, in [4] it was shown that for every such virtual knot diagram there is an embedding of the molecule in space, and there is smoothing of the vertices (recombination along the pointers) such that the resulting molecule is always unlinked. Further it was shown that the smoothing guided by the pointers differs from the existent smoothing notions defined earlier for virtual knot diagrams [56]. This opens completely new problems on virtual knot diagrams that have not been studied before.

Research on the computational nature of gene assembly is an example of genuinely interdisciplinary research that contributed to both computer science, by a whole range of novel and challenging models of computation, and to biology, by increasing our understanding of the biological nature of gene assembly – it has even led to formulating biological models of this process based on the notion of template-guided recombination.

Acknowledgments

MD and GR acknowledge support by NSF, grant 0622112. IP acknowledges support by Academy of Finland, grants 108421 and 203667. NJ has been supported in part by the NSF grants CCF 0523928 and CCF 0726396.

References

1. Alhazov A, Li C, Petre I (2009) Computing the graph-based parallel of gene assembly. *Theoret. Comput. Sci.*, in press.
2. Alhazov A, Petre I, and Rogojin V (2008) The parallel complexity of signed graphs: decidability results and an improved algorithm. *Theoret. Comput. Sci.*, in press.
3. Alhazov A, Petre I, and Rogojin V (2008) Solutions to computational problems through gene assembly. *J. Nat. Comput.* **7**(3) 385–401
4. Angeleska A, Jonoska N, Saito M (2009) DNA recombinations through assembly graphs, in review.
5. Angeleska A, Jonoska N, Saito M, Landweber LF (2007) RNA-template guided DNA assembly. *J. Theor. Biol.* **248** 706–720
6. Brijder R (2008) Gene assembly and membrane systems. PhD thesis, University of Leiden.
7. Brijder R, Hoogeboom H (2008) The fibers and range of reduction graphs in ciliates. *Acta Informatica* **45** 383–402
8. Brijder R, Hoogeboom HJ (2008) Extending the overlap graph for gene assembly in ciliates. In: C. Martín-Vide, F. Otto, and H. Fernau (Eds.) *LATA 2008*, *LNCS* **5196** 137–148
9. Brijder R, Hoogeboom H, Muskulus M (2008) Strategies of loop recombination in ciliates. *Discr. Appl. Math.* **156** 1736–1753
10. Brijder R, Hoogeboom H, Rozenberg G (2006) Reducibility of gene patterns in ciliates using the breakpoint graph. *Theoret. Comput. Sci.* **356** 26–45
11. Cavalcanti A, Clarke TH, Landweber L (2005) MDS_IES_DB: a database of macronuclear and micronuclear genes in spirotrichous ciliates. *Nucl. Acids Res.* **33** 396–398
12. Chang WJ, Bryson PD, Liang H, Shin MK, Landweber L (2005) The evolutionary origin of a complex scrambled gene. *PNAS* **102**(42) 15149–15154
13. Chang WJ, Kuo S, Landweber L. (2006) A new scrambled gene in the ciliate *Uroleptus*. *Gene*, **368** 72 – 77
14. Daley M, Domaratzki M (2007) On codes defined by bio-operations. *Theor. Comput. Sci.* **378**(1) 3–16
15. Daley M, Domaratzki M, Morris A (2007) Intramolecular template-guided recombination. *Int. J. Found. Comput. Sci.* **18**(6) 1177–1186
16. Daley M, Ibarra OH, Kari L (2003) Closure propeties and decision questions of some language classes under ciliate bio-operations. *Theoret. Comput. Sci.* **306**(1-3) 19–38
17. Daley M, Ibarra OH, Kari L, McQuillan I, Nakano K (2003) The ld and dlad bio-operations on formal languages. *J. Autom. Lang. Comb.* **8**(3) 477–498
18. Daley M, Kari L, McQuillan I (2004) Families of languages defined by ciliate bio-operations. *Theoret. Comput. Sci.* **320**(1) 51–69
19. Daley M, McQuillan I (2006) Useful templates and iterated template-guided DNA recombination in ciliates. *Theory Comput. Syst.* **39**(5) 619–633
20. Daley M, Mc Quillan I (2005) Template-guided DNA recombination, *Theoret. Comput. Sci.* **330**(2) 237–250
21. Daley M, McQuillan I (2005) On computational properties of template-guided DNA recombination. In: Carbone A, Pierce N (eds) *DNA 11 LNCS* **3892** 27–37
22. Daley M, McQuillan I, Stover N, Landweber LF (2008) A simple topological mechanism for gene descrambling in Stichotrichous ciliates. *under review*
23. Dassow J, Holzer M (2005) Language families defined by a ciliate bio-operation: hierarchies and decidability problems. *Int. J. Found. Comput. Sci.* **16** (4) 645–662

24. Dassow J, Mitrana V, Salomaa A (2002) Operations and languages generating devices suggested by the genome evolution. *Theoret. Comput. Sci.* **270**(1-2) 701–738
25. Dassow J, Vaszil G (2006) Ciliate bio-operations on finite string multisets. In: Ibarra OH, Dang Z (Eds.) *DLT 2006*, *LNCS* **4036** 168–179
26. Dennis C, Fedorov A, Käs E, Salomé L, Grigoriev M (2004) RuvAB-directed branch migration of individual Holliday junctions is impeded by sequence heterology. *The EMBO Journal* **23** 2413–2422
27. Domaratzki M (2007) Equivalence in template-guided recombination, *J. Nat. Comp.* **7**(3) 439–449
28. Ehrenfeucht A, Petre I, Prescott DM, Rozenberg G (2000) Universal and simple operations for gene assembly in ciliates. In: Mitrana V, Martin-Vide C (Eds.) *Where Mathematics, Computer Science, Linguistics and Biology Meet*, Kluwer, Dordrecht, 329–342
29. Ehrenfeucht A, Harju T, Petre I, Prescott DM, Rozenberg G (2003) *Computation in Living Cells: Gene Assembly in Ciliates*. Springer, Berlin Heidelberg NewYork
30. Ehrenfeucht A, Harju T, Petre I, Prescott DM, Rozenberg G (2003) Formal systems for gene assembly in ciliates. *Theoret Comput. Sci.*, **292**, 199–219
31. Ehrenfeucht A, Harju T, Rozenberg G (2002) Gene assembly in ciliates through circular graph decomposition. *Theoret. Comput. Sci.* **281** 325–349
32. Ehrenfeucht A, Petre I, Prescott DM, Rozenberg G (2001) Circularity and other invariants of gene assembly in ciliates. In: Ito M, Păun G and Yu S (Eds.) *Words, semigroups, and transductions*, World Scientific, Singapore 81–97
33. Ehrenfeucht A, Prescott DM, Rozenberg G (2001) Computational aspects of gene (un)scrambling in ciliates. In: Landweber LF, Winfree E (eds.) *Evolution as Computation*, Springer, Berlin Heidelberg New York 216–256
34. Ehrenfeucht A, Prescott DM, Rozenberg G (2007) A model for the origin of internal eliminated segments (IESs) and gene rearrangement in stichotrichous ciliates. *J. Theor. Biol.* **244**(1) 108-114
35. Ehrenfeucht A, Rozenberg G (2006) Covers from templates. *Int. J. Found. Comput. Sci.* **17**(2) 475–488
36. Freund R, Martin-Vide C, Mitrana V (2002) On some operations on strings suggested by gene assembly in ciliates. *New Generation Computing* **20**(3) 279–293
37. Greslin AF, Prescott DM, Oka Y, Loukin SH, Chappell JC (1989) Reordering of nine exons is necessary to form a functional actin gene in *Oxytricha nova*. *PNAS* **86**(16) 6264–6268
38. Harju T, Li C and Petre I (2008) Parallel complexity of signed graphs for gene assembly in ciliates. *Soft Computing - A Fusion of Foundations, Methodologies and Applications* **12**(8) 731–737
39. Harju T, Li C and Petre I (2008) Graph theoretic approach to parallel gene assembly. *Discr. Appl. Math.* **156**(18) 3416-3429
40. Harju T, Li C, Petre I, and Rozenberg G (2006) Parallelism in gene assembly. *Nat. Comp.* **5**(2) 203–223
41. Harju T, Li C, Petre I, and Rozenberg G (2007) Complexity measures for gene assembly. In: Tuyls K (Ed.) Proceedings of the *Knowledge Discovery and Emergent Complexity in Bioninformatics* workshop. *LNBI* **4366** 42–60
42. Harju T, Petre I, Rogojin V and Rozenberg G (2006) Simple operations for gene assembly. In: Kari L (ed.) *Proceedings of DNA-based computers 11*, *LNCS* **3892**, Springer 96–111
43. Harju T, Petre I, Rogojin V and Rozenberg G (2008) Patterns of simple gene assembly in ciliates, *Discr. Appl. Math.* **156**(14): 2581–2597

44. Harju T, Petre I and Rozenberg G (2004) Two models for gene assembly. In: Karhumäki J, Păun G, Rozenberg G (Eds) *Theory is Forever*, Springer, 89–101
45. Harju T, Petre I and Rozenberg G (2004) Gene assembly in ciliates: Formal frameworks. In: Paun G, Rozenberg G, Salomaa A (Eds.), *Current trends in Theoretical Computer Science (The Challenge of the New Century)*, World Scientific, 543–558
46. Harju T, Petre I and Rozenberg G (2006) Modelling simple operations for gene assembly. In: Chen J, Jonoska N, Rozenberg G (Eds) *Nanotechnology: Science and Computation*, Springer 361–376
47. Hausmann K, Bradbury PC (Eds.) (1997) Ciliates: Cells As Organisms. Vch Pub
48. Hoffman DC, Prescott DM (1996) The germline gene encoding DNA polymerase alpha in the hypotrichous ciliate Oxytricha nova is extremely scrambled. *Nucleic Acids Res.* **24**(17) 3337 – 3340
49. Ilie L, Solis-Oba R (2006) Strategies for DNA self-assembly in ciliates. In: C. Mao, T. Yokomori (Eds.), *Proc. of the 12th International Meeting on DNA Computing (DNA'06)*, LNCS **4287**, Springer 71–82
50. Ishdorj T-O, Loos, R, Petre I (2008) Computational efficiency of intermolecular gene assembly, *Fund. Informaticae*, **84**(3-4), 363–373
51. Ishdorj T-O, Petre I, Rogojin V (2007) Computational power of intramolecular gene assembly, *Int. J. Found. Comp. Sci.* **18**(5): 1123–1136
52. Ishdorj T-O, Petre I (2008) Gene assembly models and boolean circuits, *Int. J. Found. Comput. Sci.* **19**(5) 1133–1145
53. Jahn CL, Klobutcher LA (2000) Genome remodeling in ciliated protozoa. *Ann. Rev. Microbiol.* **56** 489–520
54. Kari L, Kari J, Landweber LF (1999) Reversible molecular computation in ciliates. In: Karhumäki J, Maurer H, Păun G, Rozenberg G (Eds.) *Jewels are Forever*, Springer, Berlin HeidelbergNew York 353–363
55. Kari L, Landweber LF (1999) Computational power of gene rearrangement. In: Winfree E, Gifford DK (Eds.) *Proceedings of DNA Bases Computers, V* American Mathematical Society 207–216
56. Kauman LH (1999) Virtual knot theory. *European J. Combinatorics* **20** 663–690.
57. Krasnow MA, Stasiak A, Spengler SJ, Dean F, Koller T, Cozzarelli NR (1983) Determination of the absolute handedness of knots and catenanes of DNA. *Nature* **304**(5926) 559–560
58. Landweber LF, Kari L (1999) The evolution of cellular computing: Nature's solution to a computational problem. In: Kari L, Rubin H, Wood D (Eds.) Special issue of *Biosystems*: *Proceedings of DNA Based Computers IV* **52**(1-3), Elsevier, Amsterdam, 3–13
59. Landweber LF, Kari L (2002) Universal molecular computation in ciliates. In: Landweber LF, Winfree E (Eds.) *Evolution as Computation*, Springer, Berlin Heidelberg New York 257–274.
60. Landweber LF, Kuo T-C, Curtis EA (2000) Evolution and assembly of an extremely scrambled gene. *Proc. Natl. Acad. Sci.* **97**(7) 3298 – 3303.
61. Langille M, Petre I (2006) Simple gene assembly is deterministic. *Fund. Inf.* **72** 1–12
62. Langille M, Petre I (2007) Sequential vs. parallel complexity in simple gene assembly. *Theoret. Comput. Sci.* **395**(1) 24–30
63. Langille M, Petre I, Rogojin V (2008) Three models for gene assembly in ciliates: a comparison. In: Leibnitz K, Steglich S (Eds) *Proceedings of Bionetics 2008*, in press.
64. Mitcham JL, Lynn AJ, Prescott DM (1992) Analysis of a scrambled gene: the gene encoding alpha-telomere-binding protein in Oxytricha nova. *Genes Dev.* **6**(5) 788–800

65. Möllenbeck M, Zhou Y, Cavalcanti ARO, Jönsson F, Higgins BP, Chang W-J, Juranek S, Doak TG, Rozenberg G, Lipps HJ, Landweber LF (2008) The pathway to detangle a scrambled gene. *PLoS ONE* **3**(6) 2330
66. Nowacki M, Vijayan V, Zhou Y, Schotanus K, Doak TG, Landweber LF (2008) RNA-mediated epigenetic programming of a genome-rearrangement pathway. *Nature* **451** 153–158
67. Petre I (2006) Invariants of gene assembly in stichotrichous ciliates. *IT*, Oldenbourg Wissenschftsverlag, **3** 161–167.
68. Petre I, Skogman S (2006) Gene assembly simulator. `http://combio.abo.fi/simulator/simulator.php`
69. Petre I, Rogojin V (2008) Decision problems for shuffled genes. *Information and Computation* **206**(11) 1346–1352
70. Pevzner P (2000) Computational Molecular Biology: An Algorithmic Approach. MIT Press
71. Prescott DM (1994) The DNA of Ciliated Protozoa. *Microbiological Reviews* **58**(2) 233–267.
72. Prescott DM (1999) The evolutionary scrambling and developmental unscrambling of germline genes in hypotrichous ciliates. *Nucleic Acids Res* **27**(5) 1243–1250
73. Prescott DM (2000) Genome gymnastics: unique modes of dna evolution and processing in ciliates. *Nat Rev Genet* **3** 191–198
74. Prescott DM, DuBois M (1996) Internal eliminated segments (IESs) of Oxytrichidae. *J. Eukariot. Microbiol.* **43** 432–441
75. Prescott DM, Ehrenfeucht A, Rozenberg G (2001) Molecular operations for DNA processing in hypotrichous ciliates. *Europ. J. Protistology* **37** 241–260
76. Prescott DM, Ehrenfeucht A, Rozenberg G (2001) Template-guided recombination for IES elimination and unscrambling of genes in stichotrichous ciliates. *J. Theor. Biol.* **222** 323–330
77. Prescott DM, Greslin AF (1992) Scrambled actin I gene in the micronucleus of Oxytricha nova. *Developmental Genetics* **13**(1) 66–74
78. Prescott DM, Murti G (2002) Topological organization of DNA molecules in the macronucleus of hypotrichous ciliated protozoa. *Chromosome Research* **10**(2) 165–173
79. Setubal J, Meidanis J (1997) Introduction to Computional Molecular Biology. PWS Publishing Company
80. White, JH (1992) Geometry and topology of DNA and DNA-protein interactins. In: Sumners et al. (Eds.) *New Scientific Applications of Geometry and Topology*, AMS 17–38
81. Yan H, Zhang X, Shen Z, Seeman NC (2002) A robust DNA mechanical device controlled by hybridization topology. *Nature* **415** 62–65
82. Yurke B, Turberfield AJ, Mills AP, Simmel Jr. FC (2000) A DNA fueled molecular machine made of DNA. *Nature* **406** 605–608
83. Zerbib D, Mezard C, George H, West SC (1998) Coordinated actions of RuvABC in Holliday junction processing. *J Mol Biol.* **281**(4) 621–630