Characterizing the micronuclear gene patterns in ciliates

Andrzej Ehrenfeucht [*] Tero Harju [†]
Ion Petre [‡] Grzegorz Rozenberg [§] [*]

Abstract

The process of *gene assembly* in ciliates is one of the most complex examples of DNA processing known in any organism, and it is fascinating from the computational point of view - it is a prime example of DNA computing *in vivo*. In this paper we continue to investigate the three molecular operations (*ld, hi*, and *dlad*) that were postulated to carry out the gene assembly process in the intramolecular fashion. In particular, we focus on the understanding of the IES/MDS patterns of micronuclear genes, which is one of the important goals of research on gene assembly in ciliates. We succeed in characterizing for each subset $\mathcal{S}$ of the three molecular operations those patterns that can be assembled using operations in $\mathcal{S}$. These results enhance our understanding of the structure of micronuclear genes (and of the nature of molecular operations). They allow one to establish both *similarity* and *complexity* measures for micronuclear genes.

1 Introduction

Interest in *bioinformatics* and *DNA computing* has been rapidly growing in the recent past, with research on the borderline between biology and computer science. In *bioinformatics*, computer science typically assists the molecular biologists in understanding the structure and functioning of biomolecules, such as DNA and proteins, in living organisms. In *DNA computing*, molecular biology assists computer scientists to achieve a really bold goal: to implement computational processes through biomolecular structures, taking advantage of the massive parallelism of the biochemical processes, and to replace (or to complement) the silicon based computers by DNA based computers. One line of research in DNA computing is concerned with the use of biomolecules *in vitro*, i.e., outside living cells (typically in test tubes). Another branch of DNA computing deals with the computational process *in vivo*, i.e., in living cells. Understanding the

[*]University of Colorado, Department of Computer Science, Boulder CO 80309-0347 USA, email:andrzej@cs.colorado.edu

[†]Department of Mathematics, University of Turku and Turku Centre for Computer Science, Turku FIN 20014 Finland, email:harju@utu.fi

[‡]Turku Centre for Computer Science and Department of Mathematics, University of Turku, FIN 20520, Finland, email:ipetre@cs.utu.fi

[§]Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands, email:rozenber@liacs.nl

in vivo computational processes contributes to our understanding of computational properties of biomolecules in their natural environment: the living cell. This area of research belongs also to bioinformatics, as it contributes to the understanding of the computational nature of complex biological processes taking place in living organisms, and their computational aspects.

Ciliates are an ancient group of organisms (their origin is estimated at about 2×10^9 years ago). It is a very rich group, comprising about 10 000 genetically different organisms. A unique feature of ciliates is their nuclear dualism. They have developed two nuclei: a germline nucleus (*micronucleus*) used in cell mating only, and a somatic nucleus (*macronucleus*), providing RNA transcripts to operate the cell. The DNA in the micronucleus is hundreds of thousands base pairs long, with genes occurring individually or in groups, dispersed along the DNA molecule, and separated by stretches of spacer DNA. The DNA molecules in the macronucleus are gene-size, on average about 2000 base pairs long. The micronuclear gene consists of gene segments, called MDSs, separated by multiple, noncoding segments called IESs. During sexual reproduction a macronucleus develops from a micronucleus, and during this process IESs are excised, and MDSs are ligated to form transcriptionally competent genes. This gene assembly process is very interesting from a computational point of view. One of the amazing features of this process is that ciliates apparently know the *linked list data structure* and use it (in the assembly of macronuclear genes) by a very elegant pattern matching process.

The computational beauty of gene assembly in ciliates has been brought to the attention of the DNA computing community by L.Landweber and L.Kari (see, e.g., [6] and [7]). The focus of their research is the computational power of molecular operations used in gene assembly. This power is measured using the yardsticks of computation theory - thus, e.g., it is shown in [7] that using two operations postulated in [6] one can simulate Turing machines.

A different line of research on the computational nature of gene assembly in ciliates has been initiated in [5] and [12]. Here one investigates the *assembly process itself*. A typical question is: "What are the assembly strategies for a given micronuclear gene?" This research has led to a formal system for gene assembly in ciliates, based on three molecular operations: *ld*, *hi*, and *dlad* (see [12]). The usefulness of this system has been demonstrated by applying it to *all* known cases of gene assembly in ciliates, obtaining in this way *one uniform* explanation of all these cases. Moreover, it has been proved in [2] that every micronuclear gene (pattern) can be assembled using the molecular operations postulated by this formal system - in this sense, this system can be applied to all the micronuclear genes that will be found in the future. The major difference between the set of two molecular operations considered by Landweber and Kari, and the set of three molecular operations considered here, is that the model based on the former is *intermolecular*, while the model based on the latter is *intramolecular*.

Formal systems for gene assembly, based on string rewriting and on graph rewriting were further investigated in [1], [2], [3], [4], providing useful technical tools for the investigation of the gene assembly process. We use some of these tools in this paper to learn more about the patterns of micronuclear genes (and about the nature of molecular operations). For each subset $\mathcal{S}$ of the set of the three molecular operations, we investigate the micronuclear gene patterns that can be assembled using $\mathcal{S}$ - as a matter of fact we succeed in obtaining full characterizations for each $\mathcal{S}$.

2 DNA molecules

We recall in this section some basic facts about DNA molecules needed in this paper. The reader is referred to [8] and [9] for more background.

DNA molecules are polymers composed of simple monomers called *nucleotides*. Each nucleotide consists of three components: a sugar molecule, a phosphate group, and a base. The sugar molecule has five attachment points (carbons), numbered $1'$ through $5'$. The phosphate group is attached to $5'$ and the base is attached to $1'$. There are 4 different bases (*adenine, cytosine, guanosine*, and *thymine*) denoted by A, C, G, T; since two nucleotides may differ only in their bases, there are four possible nucleotides, also denoted by A, C, G, T.

Two nucleotides can join together through a strong (covalent) bond which binds the $3'$ end of one with the $5'$ end of the other (through the phosphate on the $5'$ end, and the hydroxyl group on the $3'$ end). A *single stranded DNA molecule* is a strand of nucleotides connected in this way. In such a strand, the nucleotide at one end of the strand has the $5'$ attachment point free (i.e., available for bonding), while the nucleotide at the other end of the strand has the $3'$ attachment point free. Since the chemical properties of these ends are very different, a single stranded DNA molecule has polarity: one can read it from the $5'$ end to $3'$ end, or from the $3'$ end to the $5'$ end. Since the $5' - 3'$ direction is "preferred" by nature, the single stranded DNA molecules are read in the $5' - 3'$ direction. Single stranded DNA molecules are denoted by strings over the alphabet $\{A, C, T, G\}$ of nucleotides, where the left-to-right reading orientation of a string corresponds to the $5' - 3'$ polarity of the denoted molecule. Hence, for example, $TGCATC$ denotes the single stranded DNA molecule with the nucleotide C at its $3'$ end.

Nucleotides can also bind through their bases by forming a weak (hydrogen) bond. However, such a bond may be formed only between complementary nucleotides: hence between A and T, and between C and G (this is known as the *Watson-Crick complementarity*). *Double stranded DNA molecules* are formed when two single stranded, complementary DNA molecules bind through their bases. Such a bond may be formed only if the joined single stranded molecules are of opposite polarity. The same alphabet of nucleotides $\{A, C, G, T\}$ is used to denote the double stranded DNA molecules, by simply writing the strings denoting the two strands of a given DNA molecule above each other. In this notation, the upper string is in $5' - 3'$ direction, and the lower string is in the $3' - 5'$ direction. For example, $\begin{matrix} TACGA \\ ATGCT \end{matrix}$ denotes the double stranded molecule such that one of its strands is $TACGA$ and the other is $TCGTA$.

The molecule in our example is "perfect", in the sense that all nucleotides in one strand have a "matching" nucleotide in the other strand. There are also nonperfect molecules, e.g., $\begin{matrix} ACGTACAG \\ TGCAT \end{matrix}$ is such an example - it has the end CAG with missing complementary nucleotides in the other strand. Such an end is called *sticky*: other nonperfect molecules can bind to this one forming with this a sticky end.

For a single stranded DNA molecule α, the molecule that binds to it to form a perfect double stranded DNA molecule is the *inversion* of α, denoted by either $\mathbf{inv}(\alpha)$ or $\overline{\alpha}$. The inversion of α is obtained by taking the Watson-Crick complement of the mirror image of α. For example, for the single stranded DNA molecule $ATCGA$, its inversion is $TCGAT$. Note that when we write $\overline{\alpha}$ then we write it also in the $5'-3'$ direction; therefore we work with the inversion as $TCGAT$ and not as $TAGCT$.

In the notation for the double stranded DNA molecules, the upper string denotes one of the strands in the $5'-3'$ direction. Hence, the same molecule can be written in *two* ways depending on which strand is denoted by the upper string. For example, both $\begin{array}{c} ACGTC \\ TGCAG \end{array}$ and $\begin{array}{c} GACGT \\ CTGCA \end{array}$ denote the same molecule and so, we say that $\begin{array}{c} GACGT \\ CTGCA \end{array}$ is the *inversion* of $\begin{array}{c} ACGTC \\ TGCAG \end{array}$. Hence, for a double string α (representing a double stranded molecule), its inversion is obtained by exchanging the two single strings of α for each other, and then taking the mirror image of the resulting double string. Thus, $\begin{array}{r} GACGT \\ TCACTGCA \end{array}$ is the inversion of $\begin{array}{l} ACGTCACT \\ TGCAG \end{array}$.

3 Gene assembly in ciliates

The structure of a gene in a micronuclear chromosome in ciliates consists of MDSs (macronuclear destined sequences) separated by IESs (internally eliminated sequences). During sexual reproduction, the micronuclear genome is converted into the macronuclear genome. In this process, the IESs become excised, and the MDSs are spliced in the orthodox order $M_1, M_2, \ldots, M_\kappa$ suitable for transcription (see, e.g., [9], [10], and [11]). Each MDS M_i has the structure

$$M_i = \left(\begin{array}{c} p_i \\ \overline{p}_i \end{array}, \mu_i, \begin{array}{c} p_{i+1} \\ \overline{p}_{i+1} \end{array} \right),$$

except for M_1 and M_κ, which are of the form

$$M_1 = \left(\begin{array}{c} b \\ \overline{b} \end{array}, \mu_1, \begin{array}{c} p_2 \\ \overline{p}_2 \end{array} \right), \quad M_\kappa = \left(\begin{array}{c} p_\kappa \\ \overline{p}_\kappa \end{array}, \mu_\kappa, \begin{array}{c} e \\ \overline{e} \end{array} \right),$$

We refer to each double strand $\frac{p_i}{\overline{p}_i}$ as a *pointer*, and to μ_i as the *body* of M_i - we call $\frac{p_i}{\overline{p}_i}$ the *incoming pointer* of M_i, and $\frac{p_{i+1}}{\overline{p}_{i+1}}$ the *outgoing pointer* of M_i. The markers b and e (and their "complements" $\overline{b}$ and $\overline{e}$) are not pointers – they are just symbolic markers designating the locations where an incipient macronuclear DNA molecule will be excised from the macronuclear genome.

In the macronucleus, the MDSs $M_1, M_2, \ldots, M_\kappa$ are spliced together by "gluing" each M_j with M_{j+1} on the pointer $\frac{p_{j+1}}{\overline{p}_{j+1}}$.

It is argued in [5] and [12] that the gene assembly process in ciliates is accomplished through the following three molecular operations:

1. (*loop, direct repeat*)-*excision*, also called *ld-excision*, or *ld*, for short. This operation is applied to a molecule with a *direct repeat pattern* $\left(\frac{p}{\overline{p}}, \frac{p}{\overline{p}}\right)$ of a pointer $\frac{p}{\overline{p}}$, i.e., to a molecule where either the two occurrences of $\frac{p}{\overline{p}}$ are separated by an IES only, or they are on the two opposite ends of the molecule. The molecule is folded into a loop aligned by the pair of direct repeats and then, the operation

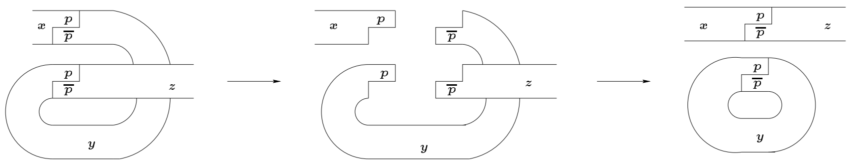

Figure 1: The *ld*-excision operation.

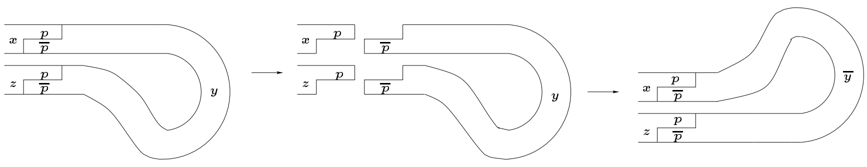

Figure 2: The *hi*-excision/reinsertion operation.

proceeds as shown in Figure 1. The excision here involves staggered cuts and the operation yields two molecules, a linear one, and a circular one. (The *ld* operation, in a somewhat simpler version, was already used in the Landweber - Kari model [6].)

An application of the *ld* operation excises a circular molecule consisting either of an IES only, or containing the whole gene in its micronuclear or intermediate form. The latter happens if the incipient or intermediate form of the macronuclear gene begins with one and ends with the other copy of a pointer X (an application of *ld* to the so positioned pointer X is called *boundary*). Once the two copies of X get into this position, the application of operations other than *ld* applied to X will not affect (the position of) the two occurrences of X. Consequently, we will assume that, unless clear otherwise, a boundary application of *ld* in a successful gene assembly is the very last step of this assembly.

2. (*hairpin, inverted repeat*)*-excision/reinsertion*, also called *hi-excision/reinsertion*, or *hi*, for short. This operation is applied to a molecule with an *inverted repeat pattern* $\left(\frac{p}{\overline{p}}, \frac{\overline{p}}{p}\right)$ of a pointer $\frac{p}{\overline{p}}$. The molecule is folded into a hairpin aligned by the inverted repeat pair and the operation proceeds as in Figure 2. The excision involves staggered cuts, but through reinsertion, the operation yields only one molecule.

3. (*double loop, alternating direct repeat*)*-excision/reinsertion*, also called *dlad-excision/reinsertion*, or *dlad*, for short. This operation is applied to a molecule with an *alternating direct repeat pattern* $\left(\frac{p}{\overline{p}}, \frac{q}{\overline{q}}, \frac{p}{\overline{p}}, \frac{q}{\overline{q}}\right)$. The molecule is folded into a double loop with one loop aligned by $\left(\frac{p}{\overline{p}}, \frac{p}{\overline{p}}\right)$, and the other by $\left(\frac{q}{\overline{q}}, \frac{q}{\overline{q}}\right)$. The operation proceeds as in Figure 3. The excision involves staggered cuts, but because of the reinsertion, the operation yields one molecule.

For more details on these molecular operations we refer the reader to [5] and [12].

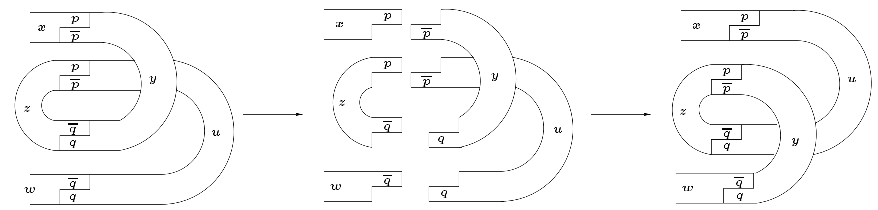

Figure 3: The *dlad*-excision/reinsertion operation.

4 Formalizing the gene assembly in ciliates

Three levels of abstraction were considered in [3], [5] and [12] in order to formalize the MDS/IES structure of genes: MDS descriptors, legal strings and legal graphs (in this order of increasing abstraction). Then three types of formal systems were investigated: MDS descriptor reduction systems (operating on MDS descriptors), string pointer reduction systems (operating on legal strings), and graph pointer reduction systems (operating on legal graphs). These formal systems were used to investigate the gene assembly process on various levels of abstraction. Surprisingly enough, it has turned out that, as far as assembly strategies are concerned, all three systems are equivalent and translatable into each other ([1]). For the purpose of this paper, it suffices to recall only the modeling of the gene assembly process through string pointer reduction system.

We simplify our notations by using positive integers $2, 3, \ldots$ and $\overline{2}, \overline{3}, \ldots$, to denote pointers (note that 1 and $\overline{1}$ are not used in our notation because each pointer links an MDS with its successor; the first pointer doing such linkage is the outgoing pointer of MDS 1 - hence, pointer 2). We obtain in this way the alphabets $\Delta = \{2, 3, \ldots\}$ and $\overline{\Delta} = \{\overline{2}, \overline{3}, \ldots\}$ and then $\Pi = \Delta \cup \overline{\Delta}$ denotes the set of pointers. We use the "bar operator" to move from Δ to $\overline{\Delta}$ and the other way around. Hence, for $p \in \Pi$, $\overline{p}$ is the *partner* of p, where $\overline{\overline{p}} = p$. For any $p \in \Pi$, we call $\{p, \overline{p}\}$ *the pointer set* of p (and of $\overline{p}$).

4.1 Preliminaries on strings

For strings u, v, we say that v is a *substring* of u if $u = w_1 v w_2$, for some strings w_1, w_2. Also, v is a *cyclic substring* of u if v is a substring of u, or $v = v_1 v_2$ and $u = v_2 w v_1$ for some strings v_1, v_2, w. We say that v is a *scattered substring* of u if there are strings $v_1, \ldots, v_n$, $u_1, \ldots, u_{n+1}$, $n \geq 1$, such that $v = v_1 \ldots v_n$ and $u = u_1 v_1 u_2 v_2 \ldots u_n v_n u_{n+1}$. A *proper* substring (cyclic, scattered substring) of u is a substring (cyclic, scattered substring, resp.) v of u such that $v \neq \Lambda$ and $v \neq u$ (where Λ denotes the empty string).

Example 1. *The string* 456 *is a substring of* 23**456**78. *Also,* 782 *is a cyclic substring of* **2**345**678**, *and* 2358 *is a scattered substring of* **23**4**5**67**8**.

A *legal string* is a nonempty string $u \in \Pi^*$ such that for each $p \in \Pi$, if p occurs in u, then u contains exactly two occurrences from the *pointer set* $\{p, \overline{p}\}$. If u contains one occurrence of p and one of $\overline{p}$, then p (and $\overline{p}$) is said to be *positive* in u, otherwise p (and $\overline{p}$) is *negative* in u. We say that u contains a pointer set $\{p, \overline{p}\}$ if u has an occurrence of p or $\overline{p}$. A legal string with no proper legal substrings is called *elementary*.

Example 2. *The string* $u = 243\overline{2}\overline{5}345$ *is legal, while* $v = 243\overline{2}\overline{5}35$ *is not legal: v has only one occurrence form the pointer set* $\{4, \overline{4}\}$. *The pointers* 2 *and* 5 *are positive in u, while the pointers* 3 *and* 4 *are negative in u. Also, u is an elementary legal string, while* $\mathbf{2\overline{5}434532}\overline{2}$ *is legal, but not elementary.*

For a string $u = x_1 x_2 \ldots x_n$, with $x_i \in \Pi$ for all $1 \leq i \leq n$, the *inverse* of u is the string $\overline{u} = \overline{x}_n \ldots \overline{x}_2 \overline{x}_1$. For a pointer $p \in \Pi$ such that $\{x_i, x_j\} \subseteq \{p, \overline{p}\}$, for some positive integers $i < j \leq n$, the *p-interval* of u is the substring $x_i x_{i+1} \ldots x_j$. Two distinct pointers $p, q \in \Pi$ *overlap in u* if the p-interval of u overlaps with the q-interval of u (recall that for $i_2 \geq i_1$, the substrings $x_{i_1} \ldots x_{j_1}$ and $x_{i_2} \ldots x_{j_2}$ of u overlap if $i_1 < i_2 < j_1 < j_2$, or $i_2 < i_1 < j_2 < j_1$). We denote by $O_u(p)$ ($O_u^+(p)$, $O_u^-(p)$, resp.) the set of (positive, negative, resp.) pointers overlapping with the pointer p in u. For technical reasons, we always include p in $O_u(p)$. If p is a positive pointer, then $p \in O_u^+(p)$, and if p is negative, then $p \in O_u^-(p)$.

Let u be a legal string and p, q two pointers in u. Then p is *connected* to q in u, if there are pointers $r_1, \ldots, r_n$, $n \geq 2$, such that $r_1 = p$, $r_n = q$, and $r_{i+1} \in O_u(r_i)$, for all i.

Example 3. *Let* $u = 2453\overline{2}\,\overline{6}\,\overline{5}73467$. *The* 3*-interval of u is the substring* $3\overline{2}\,\overline{6}\,\overline{5}73$ *and the* 5*-interval of u is* $53\overline{2}\,\overline{6}\,\overline{5}$. *The pointer* 2 *overlaps with* $3, 4$, *and* 5 *in u, but not with* 6 *and* 7. *Also,* $O_u(2) = \{2, 3, 4, 5\}$, $O_u^+(2) = \{2, 5\}$, *and* $O_u^-(2) = \{3, 4\}$. *The pointer* 2 *is connected with the pointer* 7*:* $3 \in O_u(2)$ *and* $7 \in O_u(3)$.

4.2 Micronuclear patterns

For a specific gene, the number κ of its micronuclear MDSs is fixed and we say that the gene is of *size* κ. We assume that κ is fixed for the sequel of this paper.

The process of gene assembly begins with the micronuclear precursor and it leads to a composite MDS containing all MDSs spliced in the orthodox order $M_1 M_2 \ldots M_\kappa$. For the purpose of computational analysis of the gene assembly process, one can represent a micronuclear gene (as well as the macronuclear and all the intermediate genes) by the sequences of its MDSs only (where the macronuclear gene becomes *one* composite MDS).

We use the alphabet $\Theta = \{M_{i,j}, \overline{M}_{i,j} \mid 1 \leq i \leq j \leq \kappa\}$ to denote the MDSs and their inverses. The letters $M_{i,i}$ may also be written as M_i and they denote the micronuclear MDSs, also called *elementary MDSs*. The letters $M_{i,j}$ denote the *composite MDSs* formed during the assembly process by splicing the MDSs $M_i, M_{i+1}, \ldots, M_j$. We say that an MDS sequence $M_{i_1,j_1} \ldots M_{i_n,j_n}$, $n \leq \kappa$, is *orthodox* if $i_1 = 1$, $i_l = 1 + j_{l-1}$, for all $2 \leq l \leq n$, and $j_n = \kappa$. As an example, $M_{1,3} M_{4,5} M_{6,9}$ is an orthodox sequence of size 9. Simplifying the pointer notations as in Section 3, the MDSs are represented as $M_i = (i, \mu_i, i+1)$ and $\overline{M}_i = \left(\overline{i+1}, \overline{\mu}_i, \overline{i}\right)$.

A sequence over Θ is a *micronuclear pattern* if it is a permutation of an orthodox sequence, possibly with some of its elements inverted. In its macronuclear form, an assembled gene has no pointers. On the other hand, its micronuclear precursor has pointers that flank IESs. Thus, the gene assembly process can be seen as a succession of pointer set removals by molecular operations, and so, it can be analyzed by tracing the sequences of pointers present in the intermediate molecules. Consequently, we can simplify the formal framework by denoting each MDS by the ordered pair of its pointers and markers only, i.e., if

$M = (p, \mu, q)$, then M is represented as (p, q), and its inversion $\overline{M} = (\overline{q}, \overline{\mu}, \overline{p})$ as $(\overline{q}, \overline{p})$. Moreover, the model can be further simplified by removing the parentheses from the descriptors and deleting the markers. We obtain in this way legal strings as the formalism for describing the sequences of pointers present in the micronuclear and the intermediate molecules.

Hence, for a micronuclear pattern δ, by writing the sequence of pointers and deleting the markers, we obtain the *legal string* u *corresponding to* δ. We denote this mapping by π: $\pi(\delta) = u$. Formally,

$$\pi(M_1) = 2, \quad \pi(M_\kappa) = \kappa, \quad \pi(M_i) = i(i+1), \quad \text{for } 2 < i < \kappa,$$

and $\pi(\overline{M}_i) = \overline{\pi(M_i)}$, for $1 \le i \le \kappa$. We say that the legal string u is *realistic* if there is a micronuclear pattern δ such that $u = \pi(\delta)$.

4.3 The string pointer reduction system

The *string pointer reduction system* over Π (**SPRS** for short) consists of three sets of reduction rules operating on legal strings, corresponding to the micronuclear operations of *ld, hi,* and *dlad* (in [1] and [3] these string reduction rules were denoted by **snr**, **spr**, and **sdr**, resp.).

For $p, q \in \Pi$, with $p \neq q$,

1. the **ld**-rule for p is defined by $\mathbf{ld}_p(u_1 ppu_2) = u_1 u_2$,
2. the **hi**-rule for p is defined by $\mathbf{hi}_p(u_1 pu_2 \overline{p} u_3) = u_1 \overline{u}_2 u_3$, and
3. the **dlad** rule for p, q is defined by $\mathbf{dlad}_{p,q}(u_1 pu_2 qu_3 pu_4 qu_5) = u_1 u_4 u_3 u_2 u_5$.

Let $\mathcal{LD}$ be the set of **ld**-operations, $\mathcal{LD} = \{\mathbf{ld}_p \mid p \in \Pi\}$, $\mathcal{HI}$ the set of **hi**-operations, $\mathcal{HI} = \{\mathbf{hi}_p \mid p \in \Pi\}$, $\mathcal{DLAD}$ the set of **dlad**-operations, $\mathcal{DLAD} = \{\mathbf{dlad}_{p,q} \mid p, q \in \Pi, p \neq q\}$.

Let $\mathcal{S} \subseteq \{\mathcal{LD}, \mathcal{HI}, \mathcal{DLAD}\}$ be a subset of string reduction rules. For a legal string u and a sequence of reduction rules $\rho_1, \ldots, \rho_l$ from $\mathcal{S}$, $D = (u; \rho_1, \ldots, \rho_l)$ is a *reduction of* u *by* $\rho_1, \ldots, \rho_l$ *in* $\mathcal{S}$, if ρ_1 is applicable to u, and ρ_i is applicable to $\rho_{i-1} \ldots \rho_1(u)$, for all $1 < i \le l$. We say that D is *successful* in $\mathcal{S}$ if $\rho_l \ldots \rho_1(u) = \Lambda$ - we also say then that u is *successful in* $\mathcal{S}$. Note that in the above $\rho_i \neq \rho_j$, whenever $i \neq j$. The notion of a successful assembled micronuclear pattern is defined similarly for the set of molecular operations *ld, hi,* and *dlad.*

The following is an informal version of a result from [1] and [3].

Theorem 1. *(1) Let τ be a micronuclear gene. Each sequence ζ of molecular operations that leads to a successful assembly of τ into its macronuclear form is uniquely translatable into the sequence of operations from* **SPRS** *which together with the legal string representing τ forms a successful reduction in* **SPRS***.*

(2) Let π be a legal string representing a micronuclear gene τ. Each sequence ρ of operations from **SPRS** *such that (π, ρ) is a successful reduction in* **SPRS** *is uniquely translatable into the sequence ζ of molecular operations that leads to a successful assembly of τ into its macronuclear form.*

Although the essential information about the gene assembly process is preserved in the abstraction process, some structural information about the molecule is lost. We characterize in the following those micronuclear patterns having the same corresponding realistic string.

Let $M_1M_2\dots M_\kappa$ be an orthodox sequence of size κ. Let k and t be integers such that $t \in \{0,2\}$ and $\kappa = 2k + \max\{1,t\}$, and denote $M_{\mathrm{odd}} = M_1M_3\dots M_{2k+1}$ and $M_{\mathrm{even}} = M_2M_4\dots M_{2k+t}$.

Lemma 2. *Let δ_1 and δ_2 be any sequences of MDSs such that $\pi(\delta_1) = \pi(\delta_2)$ and for all proper prefixes δ_1' of δ_1 and δ_2' of δ_2, $\pi(\delta_1') \neq \pi(\delta_2')$. Then $\{\delta_1, \delta_2\}$ equals to either $\{M_{\mathrm{odd}}, M_{\mathrm{even}}\}$ or $\{\overline{M}_{\mathrm{odd}}, \overline{M}_{\mathrm{even}}\}$*

Proof. By the hypothesis, $\delta_1 = N_1\delta_{11}$ and $\delta_2 = N_2\delta_{21}$, where $N_1 \neq N_2$. Now, $\pi(\delta_1) = \pi(\delta_2)$ implies that $\pi(N_1)$ or $\pi(N_2)$ is a proper prefix of the other. By symmetry, we can assume that $\pi(N_1)$ is a prefix of $\pi(N_2)$. By the definition of π, we have that either

(i) $N_1 = M_1$ and $N_2 = M_2$ so that $\pi(N_1) = 2$ and $\pi(N_2) = 23$, or

(i) $N_1 = \overline{M}_\kappa$ and $N_2 = \overline{M}_{\kappa-1}$ so that $\pi(N_1) = \overline{\kappa}$ and $\pi(N_2) = \overline{\kappa}\,(\overline{\kappa - 1})$.

In Case (i), $\pi(\delta_1) = 2\pi(\delta_{11}) = 23\pi(\delta_{21})$, and thus necessarily δ_{11} begins with M_3 (since $\pi(M_3) = 34$), and then δ_{21} begins with M_4. It follows inductively that $\delta_1 = M_1\delta_{11} = M_{\mathrm{odd}}$ and $\delta_2 = M_2\delta_{21} = M_{\mathrm{even}}$. Case (ii) reduces to Case (i), since $\overline{\delta}_1$ and $\overline{\delta}_2$ necessarily satisfy (i). □

The following result is then a simple consequence of Lemma 2.

Theorem 3. *Let δ_1, δ_2 be two distinct micronuclear patterns of size κ. Let $(k_1, k_2) = (\kappa, \kappa - 1)$ if κ is odd, and $(k_1, k_2) = (\kappa - 1, \kappa)$ if κ is even. Also, denote $\delta_{odd} = M_1M_3\dots M_{k_1}$ and $\delta_{even} = M_2M_4\dots M_{k_2}$. Then $\pi(\delta_1) = \pi(\delta_2)$ if and only if one of the following four conditions holds: (i) $\delta_1 = \delta_{odd}\delta_{even}$ and $\delta_2 = \delta_{even}\delta_{odd}$, or (ii) $\delta_1 = \delta_{odd}\overline{\delta}_{even}$ and $\delta_2 = \delta_{even}\overline{\delta}_{odd}$, or (iii) $\delta_1 = \overline{\delta}_{odd}\delta_{even}$ and $\delta_2 = \overline{\delta}_{even}\delta_{odd}$, or (iv) $\delta_1 = \overline{\delta}_{odd}\overline{\delta}_{even}$ and $\delta_2 = \overline{\delta}_{even}\overline{\delta}_{odd}$.*

Thus, e.g., for $\delta_1 = M_1M_3M_5\overline{M}_4\overline{M}_2$ and $\delta_2 = M_2M_4\overline{M}_5\overline{M}_3\overline{M}_1$, we have indeed $\pi(\delta_1) = \pi(\delta)$, because $\delta_1 = \delta_{\mathrm{odd}}\overline{\delta}_{\mathrm{even}}$ and $\delta_2 = \delta_{\mathrm{even}}\overline{\delta}_{\mathrm{odd}}$, where $\delta_{\mathrm{even}} = M_2M_4$ and $\delta_{\mathrm{odd}} = M_1M_3M_5$.

5 Successful patterns for realistic strings

A central question for understanding the role of the set $\{\mathbf{ld}, \mathbf{hi}, \mathbf{dlad}\}$ of molecular operations in the process of assembly is the following. What kind of micronuclear patterns can be successfully assembled by various subsets of this set? In this section, we provide a complete answer to this question: for each subset of operations $\mathcal{S}$, we characterize the realistic strings successful in **SPRS** using $\mathcal{S}$. In particular, this enables us to introduce both complexity and similarity measures, discussed in Section 6.

5.1 Preliminaries

We first consider the set of legal strings having legal substrings, reducing the problem in this case to two smaller instances of the same problem.

Lemma 4. *Let $u = \alpha v \beta$ be a legal string such that v is also a legal string. Then u is successful if and only if v and $\alpha\beta$ are successful strings.*

Let $\delta = N_1 N_2 \dots N_\kappa$ be a micronuclear pattern of size κ, where $N_i \in \{M_{l_i}, \overline{M}_{l_i}\}$. We denote the IESs interlacing the MDSs of δ by the pair of pointers or markers bordering them. If I_l is the IES separating the MDS N_l of δ from the MDS N_{l+1}, $1 \leq l < \kappa$, then δ is uniquely determined by its IES sequence $I_1 I_2 \dots I_{\kappa-1} I_\kappa$, where I_κ is the pair formed by the right pointer of N_κ and the left one of N_1. I_κ is called the *connector* of δ in [4] and it is used there to establish some invariants of the (nondeterministic process of) gene assembly.

Example 4. *Let $\delta = M_1 \overline{M}_3 M_4 M_2$ be a micronuclear pattern of size 4. If we denote each MDS by the pair of pointers or markers bordering it, then $\delta = (b,2)(\overline{4},\overline{3})(4,e)(2,3)$. Its IES sequence is $(2,\overline{4})(\overline{3},4)(e,2)(3,b)$, uniquely identifying δ.*

Let δ be a micronuclear pattern and let ι be its IES sequence. We say that ι (as well as δ) has an *IES cycle* if there is a *proper* scattered subsequence ι' of ι, such that for any pointer p occurring in ι', there are two occurrences from $\{p, \overline{p}\}$ in ι', and if a marker occurs in ι', then both markers occur in ι'. We say in this case that ι' is an *IES cycle* of ι (and of δ).

Example 5. *The micronuclear pattern in Example 4 has a linear structure. The micronuclear pattern $\delta' = M_1 M_4 \overline{M}_2 \overline{M}_5 M_7 M_3 M_6$ of size 7 has a circular structure: its IES sequence is $(2,4)(5,\overline{3})(\overline{2},\overline{6})(\overline{5},7)(e,3)(4,6)(7,b)$; its scattered subsequence $(2,4)(\overline{2},\overline{6})(4,6)$ is circular. Observe that the rest of the IES sequence, $(5,\overline{3})(\overline{5},7)(e,3)(7,b)$ is also circular.*

The property of a micronuclear pattern of having an IES cycle turns out to be instrumental in the sequel, in characterizing the micronuclear patterns that can be assembled using a subset of operations not containing the **ld**-operation. This property of micronuclear patterns can be translated to the level of realistic strings using the notions of *legal* set of strings and *disjoint cycle*, defined in the following.

A set $P = \{\alpha_1, \dots, \alpha_m\}$, $m \geq 1$, of strings over Π is *legal* if $\alpha_P = \alpha_1 \dots \alpha_m$ is legal. Note that if α_P is legal, then so is $\alpha_{i_1} \dots \alpha_{i_m}$, for any permutation $i_1, \dots, i_m$ of $1, \dots, m$. A pointer p is said to *occur* in P if it occurs in α_P. Moreover, p is *negative (positive, resp.) in P*, if it is negative (positive, resp.) in α_P.

If u is a string of length 2, $u = pq$, then p is *left* in u and q is *right* in u.

A legal set $P = \{\alpha_1, \dots, \alpha_m\}$ is a *disjoint cycle* if the following conditions are satisfied for some pointers $p_1, \dots, p_m$:

1. $|\alpha_i| = 2$, for all $1 \leq i \leq m$;
2. if $m = 1$, then $P = \{p_1 p_1\}$;
3. if $m > 1$, then $\alpha_1 = p_1 p_2$, and for each $2 \leq i \leq m$, α_i contains one occurrence from $\{p_i, \overline{p}_i\}$ and one occurrence of p_{i+1}; moreover $p_{m+1} \in \{p_1, \overline{p}_1\}$;
4. for each $2 \leq i \leq m$, if p_i is negative in P, then p_i occurs in both α_{i-1} and α_i; moreover, p_i is left in α_i if and only if p_i is right in α_{i-1};
5. for each $2 \leq i \leq m$, if p_i is positive in P, then $\overline{p}_i$ occurs in α_i and p_i occurs in α_{i-1}; moreover, $\overline{p}_i$ is left in α_i if and only if p_i is left in α_{i-1}.

Clearly, if all pointers of P are negative (in P), then P is a disjoint cycle if and only if $P = \{p_1p_2, p_2p_3, \ldots, p_mp_1\}$, for some $p_1, p_2, \ldots, p_m \in \Pi$, $m \geq 1$.

Let $P = \{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ be a disjoint cycle and u a legal string. We say that u *contains* P if all strings of P are *nonoverlapping* cyclic substrings of u. If u does not contain P, we say that u *avoids* P.

Example 6. *The sets* $P_1 = \{23, 35, 54, 42\}$ *and* $P_2 = \{2\overline{4}, \overline{5}4, 56, 62\}$ *are disjoint cycles, but the set* $P_3 = \{23, 34, \overline{4}\,\overline{2}\}$ *is not. The string* $u = 5236425463$ *contains* P_1*, and avoids* P_2*.*

Lemma 5. *Let* δ *be a micronuclear pattern and let* $u = \pi(\delta)$*. Then* u *contains a disjoint cycle if and only if* δ *has an IES cycle.*

Proof. Let ι be the IES sequence of δ. Assume first that δ has an IES cycle ι'. If the markers do not occur in ι', then the set of strings $P = \{pq \mid (p, q)$ occurs in $\iota'\}$ is a disjoint cycle of u. On the other hand, if the markers occur in ι', then the IESs not occurring in ι' form another IES cycle of ι and so, the set $P' = \{pq \mid (p, q)$ does not occur in $\iota'\}$ is a disjoint cycle of u.

Assume now that u contains a disjoint cycle P. If there is a string $pq \in P$ and an MDS M of δ such that $\pi(M) = pq$, then

$$\begin{aligned} P = \{r_1r_2 \mid \exists\, 2 \leq i < \kappa :\ (r_1, r_2) = (i, i+1) \text{ or } (r_1, r_2) = (\overline{i+1}, \overline{i})\} \\ \cup \{r_1r_2 \mid (r_1, r_2) = (\kappa, 2) \text{ or } (r_1, r_2) = (\overline{2}, \overline{\kappa})\}. \end{aligned}$$

Consequently, either $M_\kappa M_1$, or $\overline{M}_1\overline{M}_\kappa$ is a subsequence of δ and so, either (e, b), or $(\overline{b}, \overline{e})$ is an IES of δ. Since both of them are circular, δ has an IES cycle.

If there is no string $pq \in P$ such that $\pi(M) = pq$, for some MDS M of δ, then necessarily, the IESs in the set $\{(p, q) \mid pq \in P\}$ form a cycle of δ. $\square$

5.2 ld-successful patterns

Theorem 6. *A realistic string* u *corresponding to a micronuclear pattern of size* κ *has a successful reduction in* $\mathfrak{LD}$ *if and only if* $u \in \mathfrak{L} \cup \overline{\mathfrak{L}}$*, where*

$$\mathfrak{L} = \{2233\ldots\kappa\kappa\} \cup \{p(p+1)(p+1)\ldots\kappa\kappa 22\ldots(p-1)(p-1)p \mid 2 \leq p \leq \kappa\}.$$

Proof. The condition of the theorem is clearly sufficient. Consider now a realistic string u, successful in $\mathfrak{LD}$. Thus, all pointers of u are negative, i.e., $u \in \Delta^*$, or $u \in \overline{\Delta}^*$. Clearly, if u is successful in $\mathfrak{LD}$, then so is $\overline{u}$ and therefore, without loss of generality, we can assume that $u \in \Delta^*$.

As u is realistic, each pointer p has two occurrences in u, that is, $u = \alpha_p p \beta_p p \gamma_p$ for some strings $\alpha_p, \beta_p, \gamma_p$. In the string u no two pointers overlap, since an application of the (*ld*)-operation can never resolve overlapping of pointers. It follows then that if, for some pointer q, one occurrence of q occurs in β_p, then so does the other one.

If $\beta_p = \Lambda$, for all pointers p, then necessarily $u = 2233\ldots\kappa\kappa$.

Suppose that there is a pointer p such that β_p is nonempty and let q be a pointer occurring in β_p, $2 \leq q \leq \kappa$. To make a choice, let us assume that $q < p$ (the other case is solved similarly). Since u is realistic, also both $q-1$ (if $q > 2$) and $q+1$ (if $q < \kappa$) occur in β_p, in the substrings $(q-1)q$ and $q(q+1)$, respectively. Consequently, all pointers r, $2 \leq r \leq p-1$ occur in β_p.

In particular, one occurrence of $p-1$ occurs in β_p in the substring $(p-1)p$. Thus, also $p+1$ occurs in β_p in the substring $p(p+1)$ (if $p < \kappa$). Consequently, all pointers r, $p < r \leq \kappa$ occur in β_p. It thus follows that p is unique with the property that β_p is nonempty and so, β_p is a catenation of the strings qq, for all q with $q \neq p$. As u is realistic, we conclude that $u = p(p+1)(p+1)(p+2)\dots(\kappa-1)\kappa\kappa 223\dots(p-1)p$. □

Theorem 6 shows that the micronuclear patterns successful in $\mathcal{LD}$ correspond to the cyclic shifts of the orthodox sequences. Thus, the *R1* gene of the *O.nova* ([11]), described by the pattern $M_1M_2M_3M_4M_5M_6$, can be successfully assembled using $\mathcal{LD}$ only, while $M_1M_3M_2$ cannot be assembled in this way.

5.3 $\{\mathbf{ld}, \mathbf{hi}\}$-successful patterns

Lemma 7. *Let u be an elementary legal string.*

(i) If $|u| \leq 2$, then u is successful in $\mathcal{LD} \cup \mathcal{HI}$;

(ii) If $|u| > 2$, then u is successful in $\mathcal{LD} \cup \mathcal{HI}$ if and only if u contains at least one positive pointer.

Proof. The condition in the theorem is obviously necessary. Let us assume that it is not sufficient. Let u be a minimal elementary legal string having positive pointers, but unsuccessful in $\mathcal{LD} \cup \mathcal{HI}$. Consider the set S of positive pointers of u, having maximal number of overlapping negative pointers, and let $S' \subseteq S$ consist of pointers with minimal number of positive overlapping pointers. Let $p \in S'$ be the pointer occurring first in u (from left to right). In particular, $\mathbf{hi}_p$ is applicable to u. Since u is a minimal counterexample, by Lemma 4, $\mathbf{hi}_p(u)$ has a negative legal substring α. We will prove now that α is reducible in $\mathcal{LD}$.

Let A be the set of pointers in α in $\mathbf{hi}_p(u)$. Since u has no negative legal substrings, α contains a pointer $q \in O_u^+(p)$. Then $O_u^-(p) \subseteq O_u^-(q)$, as otherwise α would contain a positive pointer. Since $|O_u^-(p)|$ is maximal, $O_u^-(p) = O_u^-(q)$. By the same argument, also $O_u^+(q) \subseteq O_u^+(p)$ and so, $O_u^+(p) = O_u^+(q)$. Thus, $O_u(p) = O_u(q)$, for all $q \in O_u^+(p)$, q occurring in α in $\mathbf{hi}_p(u)$.

Clearly, $A \cap O_u(p) \subseteq O_u^+(p)$. By the above, for any $q \in A \cap O_u(p)$, $O_u(q) = O_u(p)$. On the other hand, since u has no legal substrings, any two pointers of u are connected in u. Thus, there are no pointers $r \in A \setminus O_u(p)$. Consequently, $A \subseteq O_u^+(p)$ in u. Let $A = \{q_1, \dots, q_i\}$, for some $i \geq 1$. Since $O_u(p) = O_u(q_k)$, for all $1 \leq k \leq i$, and u has no legal substrings, it follows that $u = u_1pq_1\dots q_iu_2\overline{p}\overline{q}_1\dots\overline{q}_iu_3$, for some u_1, u_2, u_3. Thus, any negative legal substring of $\mathbf{hi}_p(u)$ is of the form $\overline{q}_i\dots\overline{q}_1\overline{q}_1\dots\overline{q}_i$, for some $i \geq 1$. Consequently, any negative legal substring of $\mathbf{hi}_p(u)$ is successful in $\mathcal{LD}$ and thus, also in $\mathcal{LD} \cup \mathcal{HI}$. However, $\mathbf{hi}_p(u)$ may have nonnegative legal substrings. As u was a minimal counterexample, all such substrings are successful in $\mathcal{LD} \cup \mathcal{HI}$. This is a contradiction, as it implies that u is successful in $\mathcal{LD} \cup \mathcal{HI}$. □

The following result follows then by Lemma 4 and Lemma 7.

Theorem 8. *A realistic word u has a successful reduction in $\mathcal{LD} \cup \mathcal{HI}$ if and only if for all legal substrings v of u, if $v = v_1u_1v_2\dots v_ju_jv_{j+1}$, where each u_i is a legal substring, then $v_1v_2\dots v_{j+1}$ contains a positive pointer or it is reducible in $\mathcal{LD}$.*

Thus, for instance, the pattern $M_1\overline{M}_2M_3$ has a successful reduction in $\mathcal{LD}\cup\mathcal{HI}$, but no successful reductions in either $\mathcal{LD}$, or $\mathcal{HI}$. On the other hand, the pattern $M_3M_1M_7M_4\overline{M}_5M_6M_9M_8M_2$, described by the realistic string $u = 3427845\overline{6}\,\overline{5}6798923$, has no successful reduction in $\mathcal{LD}\cup\mathcal{HI}$, since if we choose $v = u$ and consider the decomposition $v = v_1u_1v_2$, where $v_1 = 342784$, $u_1 = 5\overline{6}\,\overline{5}6$ and $v_2 = 798923$, then v_1v_2 has no positive pointers and it is not reducible in $\mathcal{LD}$.

5.4 $\{\mathbf{ld}, \mathbf{dlad}\}$-successful patterns

Theorem 9. *A realistic string u has a successful reduction in $\mathcal{LD}\cup\mathcal{DLAD}$ if and only if all the pointers in u are negative.*

Proof. The condition in the theorem is clearly necessary since both **dlad** and **ld** reduce only negative pointers, and neither of them changes the sign of any of the pointers. Consider now an elementary string u, consisting of negative pointers only. If there are two overlapping pointers p and q, then $\mathbf{dlad}_{p,q}$ is applicable to u. On the other hand, if there are no overlapping pointers in u, then u is reducible to Λ in $\mathcal{LD}$. The claim follows from Lemma 4 $\square$

Thus, the α-*TP* gene of *O.nova* ([11]), described by the pattern $M_1M_3M_5M_7$ $M_9M_{11}M_2M_4M_6M_8M_{10}M_{12}M_{13}M_{14}$ has a successful reduction in $\mathcal{LD}\cup\mathcal{DLAD}$, but it cannot be assembled in either $\mathcal{LD}$, or $\mathcal{DLAD}$. On the other hand, the pattern $M_1\overline{M}_2M_3$ cannot by assembled using $\mathcal{LD}\cup\mathcal{DLAD}$ only since it is described by the realistic string $23\overline{3}\,\overline{2}34$ which does have positive pointers (2 and 3).

As a consequence of Theorems 8 and 9, we obtain the universality of the set $\{\mathbf{ld}, \mathbf{hi}, \mathbf{dlad}\}$ of operations, a result already known from [2].

Corollary 10. *(*$\{\mathbf{ld}, \mathbf{hi}, \mathbf{dlad}\}$**-successful patterns***) Any realistic string has a successful reduction in $\mathcal{LD}\cup\mathcal{HI}\cup\mathcal{DLAD}$. Moreover, any elementary realistic string can be successfully assembled using only two operations: either* **ld** *and* **hi***, or* **ld** *and* **dlad**.

The last claim of Corollary 10 is not true for all micronuclear genes. E.g., the *actin I* gene of *O.trifallax* ([11]), described by the pattern $M_3M_4M_6M_5M_7M_9$ $M_{10}\overline{M}_2M_1M_8$ and by the realistic string $3\,4\,4\,5\,6\,7\,5\,6\,7\,8\,9\,10\,10\,\overline{3}\,\overline{2}\,2\,8\,9$, cannot be assembled either in $\mathcal{LD}\cup\mathcal{HI}$ (because of the legal substring 567567), or in $\mathcal{LD}\cup\mathcal{DLAD}$ (because of the positive pointers). Nevertheless, it can be successfully assembled in $\mathcal{LD}\cup\mathcal{HI}\cup\mathcal{DLAD}$. The same holds for the *actin I* gene of *O.nova*, given by the pattern $M_3M_4M_6M_5M_7M_9\overline{M}_2M_1M_8$ and by the realistic string $3\,4\,4\,5\,6\,7\,5\,6\,7\,8\,9\,\overline{3}\,\overline{2}\,2\,8\,9$.

5.5 dlad-successful patterns

Theorem 11. *A realistic string u has a successful reduction in $\mathcal{DLAD}$ if and only if u consists of negative pointers only and it avoids disjoint cycles.*

Proof. Let u be a realistic string having a successful reduction in $\mathcal{DLAD}$. Clearly, u consists of negative pointers only. Assume that u contains a disjoint cycle P. Consider two pointers p, q such that $\mathbf{dlad}_{p,q}$ is applicable to u, i.e., $u = u_1pu_2qu_3pu_4qu_5$, for some u_i. We claim that $v = \mathbf{dlad}_{p,q}(u)$ also contains a disjoint cycle.

If p and q do not occur in P, then obviously v contains P. If only one of them, say p, occurs in P, then there are pointers $r, s \notin \{p, q\}$ such that $rp, ps \in P$. Thus, $v = u_1u_4u_3u_2u_5$ contains the disjoint cycle $Q = (P \setminus \{rp, ps\}) \cup \{rs\}$.

If both p and q occur in P, then we have three possibilities: (i) $pq, qp \notin P$, or (ii) $pq \in P$ and $qp \notin P$, or (iii) $qp \in P$ and $pq \notin P$.

In Case (i), there are pointers r_1, r_2, s_1, s_2 such that $r_1p, pr_2, s_1q, qs_2 \in P$. Then v contains the disjoint cycle $(P \setminus \{r_1p, pr_2, s_1q, qs_2\}) \cup \{r_1r_2, s_1s_2\}$.

In Case (ii), there are pointers r, s such that $rp, pq, qs \in P$. Then v contains the disjoint cycle $(P \setminus \{rp, pq, qs\}) \cup \{rs\}$.

In Case (iii), there are pointers r, s such that $rq, qp, ps \in P$. Then v contains the disjoint cycle $(P \setminus \{rq, qp, ps\}) \cup \{rs\}$.

Consequently, there must be realistic strings of length four containing a disjoint cycle and having a successful reduction in $\mathcal{DLAD}$. This is a contradiction.

To prove the reverse implication, assume that there are realistic strings avoiding any disjoint cycles, but unsuccessful in $\mathcal{DLAD}$. Let u be a minimal such string with respect to length, $u = \pi(\delta)$, for a micronuclear pattern δ. Clearly, if **dlad** is not applicable to u on any pair of pointers, then u contains at least the disjoint cycle $\{22\}$, which is impossible. Thus, there are some pointers p, q such that $\mathbf{dlad}_{p,q}$ is applicable to u. Moreover, since u is a minimal counterexample, the string $v = \mathbf{dlad}_{p,q}(u)$ contains a disjoint cycle Q. Clearly, v is a realistic string, $v = \pi(\delta')$, where $\delta' = \mathbf{dlad}_{p,q}(\delta)$. The pointers p and q can overlap in δ in several ways, as follows:

$$\begin{array}{ll}
(a) & \delta = \delta_1(p, r_1)\delta_2(q, r_2)\delta_3(r_3, p)\delta_4(r_4, q)\delta_5 \\
(b) & \delta = \delta_1(p, r_1)\delta_2(q, p)\delta_4(r_4, q)\delta_5 \\
(c) & \delta = \delta_1(p, r_1)\delta_2(r_2, q)\delta_3(r_3, p)\delta_4(q, r_4)\delta_5 \\
(d) & \delta = \delta_1(p, q)\delta_3(r_3, p)\delta_4(q, r_4)\delta_5 \\
(e) & \delta = \delta_1(r_1, p)\delta_2(q, r_2)\delta_3(p, r_3)\delta_4(r_4, q)\delta_5 \\
(f) & \delta = \delta_1(r_1, p)\delta_2(q, r_2)\delta_3(p, q)\delta_5 \\
(g) & \delta = \delta_1(r_1, p)\delta_2(r_2, q)\delta_3(p, r_3)\delta_4(q, r_4)\delta_5
\end{array}$$

for some MDS sequences δ_i. In each case, one can prove that if v contains a disjoint cycle, then so does u. We prove it in details only for Case (e), as the other cases can be proved analogously.

In Case (e),

$$\begin{aligned}
\delta &= \delta_1(r_1, p)\delta_2(q, r_2)\delta_3(p, r_3)\delta_4(r_4, q)\delta_5 \quad \text{and} \\
\delta' &= \delta_1(r_1, r_3)\delta_4(r_4, r_2)\delta_3\delta_2\delta_5.
\end{aligned}$$

Since v contains a disjoint cycle, by Lemma 5, δ' has an IES cycle ι'; let $\mathcal{I}'$ be the set of IESs (of δ') occurring in ι'. Since u contains no disjoint cycle, δ does not have IES cycles and so, $\mathcal{I}'$ must contain some IESs of δ' which do not occur in δ.

If $\delta_2, \delta_3, \delta_5$ are nonempty, let $\delta_i = (p_i, q_i)\delta_i'(p_i', q_i')$, for $i \in \{2, 3, 5\}$ (including the possibility that $\delta_i' = \Lambda$ and $\delta_i = (p_i, q_i) = (p_i', q_i')$). Also, if $\delta_1 \neq \Lambda$, then let $\delta_1 = (p_1, q_1)\delta_1'$, otherwise let $p_1 = r_1$.

We distinguish now the following three main cases, depending on how many of $\delta_2, \delta_3, \delta_5$ are empty MDS sequences. In each of these cases, we can build a set $\mathcal{I}$ such that the IESs of $\mathcal{I}$ form a cycle of δ, leading to contradiction.

(i) If $\delta_2, \delta_3, \delta_5$ are all nonempty, then (q_3', p_2) and (q_2', p_5) are the only IESs of δ' not occurring in δ. If $(q_3', p_2) \in \mathcal{I}'$, then we replace it in $\mathcal{I}$ by the IESs (q_3', p) and $(p, p_2)\}$. If $(q_2', p_5) \in \mathcal{I}'$, then we replace it in $\mathcal{I}$ by (q_2', q) and $(q, p_5)\}$. In both cases, the IESs in $\mathcal{I}$ form a cycle of δ, which is a contradiction.

(ii) If only one of $\delta_2, \delta_3, \delta_5$ is empty, then we have the following subcases:

(ii.1) If $\delta_2 = \Lambda$, then the IES occurring in δ' but not in δ is (q_3', p_5). The $\mathcal{I} = (\mathcal{I}' \setminus \{(q_3', p_5)\}) \cup \{(q_3', p), (p, q), (q, p_5)\}$ (observe that (p, q) is indeed an IES of δ in this case).

(ii.2) If $\delta_3 = \Lambda$, then the IESs occurring in δ' but not in δ are (r_2, p_2) and (q_2', p_5). If $(r_2, p_2) \in \mathcal{I}'$, then we replace it in $\mathcal{I}$ by the IESs (r_2, p) and (p, p_2). If $(q_2', p_5) \in \mathcal{I}'$, then we replace it in $\mathcal{I}$ by the IESs (q_2', q) and (q, p_5).

(ii.3) If $\delta_5 = \Lambda$, then the IESs occurring in δ' but not in δ are (q_3', p_2) and (q_2', p_1). If $(q_3', p_2) \in \mathcal{I}'$, then we replace it in $\mathcal{I}$ by the IESs (q_3', p) and (p, p_2). If $(q_2', p_1) \in \mathcal{I}'$, then we replace it in $\mathcal{I}$ by the IESs (q_2', q) and (q, p_1).

(iii) If two of $\delta_2, \delta_3, \delta_5$ are empty, then we have the following subcases:

(iii.1) If $\delta_2 = \delta_3 = \Lambda$, then the IES occurring in δ' but not in δ is (r_2, p_5). Then $\mathcal{I} = (\mathcal{I}' \setminus \{(r_2, p_5)\}) \cup \{(r_2, p), (p, q), (q, p_5)\}$.

(iii.2) If $\delta_2 = \delta_5 = \Lambda$, then the IES occurring in δ' but not in δ is (q_3', p_1). Then $\mathcal{I} = (\mathcal{I}' \setminus \{(q_3', p_1)\}) \cup \{(q_3', p), (p, q), (q, p_1)\}$.

(iii.3) If $\delta_3 = \delta_5 = \Lambda$, then the IESs occurring in δ' but not in δ are (r_2, p_2) and (q_2', p_1). If $(r_2, p_2) \in \mathcal{I}'$, then we replace it in $\mathcal{I}$ by the IESs (r_2, p) and (p, p_2). If $(q_2', p_1) \in \mathcal{I}'$, then we replace it in $\mathcal{I}$ by the IESs (q_2', q) and (q, p_1).

(iv) If $\delta_2 = \delta_3 = \delta_5 = \Lambda$, then the only IES of δ' not occurring in δ is (r_2, p_1) and $\mathcal{I} = (\mathcal{I}' \setminus \{(r_2, p_1)\}) \cup \{(r_2, p), (p, q), (q, p_1)\}$.

Consequently, v contains no disjoint cycle and thus, since $|v| < |u|$, v is successful in $\mathcal{DLAD}$. Since $u = \mathbf{dlad}_{p,q}(u)$, it follows that u is also successful in $\mathcal{DLAD}$; a contradiction. $\square$

According to Theorem 11, the pattern $M_2 M_1 M_3$ has a successful reduction in $\mathcal{DLAD}$, while the pattern $M_1 M_2 M_3$ has no such reduction in $\mathcal{DLAD}$.

5.6 hi-successful patterns

Lemma 12. *An elementary legal string has a successful reduction in $\mathcal{HI}$ if and only if it contains at least one positive pointer and avoids disjoint cycles.*

Proof. Let u be an elementary string successful in $\mathcal{HI}$, $u = \pi(\delta)$, for some realistic MDS descriptor δ. Also, let ι be its IES sequence. Necessarily, u has at least one positive pointer. Let us assume that u contains a disjoint cycle. Then, by Lemma 5, δ has an IES cycle ι'. Let p be an arbitrary positive pointer of u. If p does not occur in ι', then ι' is also an IES cycle of $\mathbf{hi}_p(u)$. If p does occur in ι', then there are two pointers q and r such that either (p, q) and $(\overline{p}, r)$, or (q, p) and $(r, \overline{p})$ occur in ι'. In the first case, replacing the IESs (p, q) and $(\overline{p}, r)$ with $(\overline{q}, r)$ we obtain an IES cycle of $\mathbf{hi}_p(u)$. In the second case, we obtain again an IES cycle of $\mathbf{hi}_p(u)$ by replacing in ι' the IESs (q, p) and $(r, \overline{p})$ with $\{\overline{q}, r)$. Consequently, by Lemma 5, $\mathbf{hi}_p(u)$ contains a disjoint cycle, for any positive pointer p of u. Thus, there must be strings v of length two, successful in $\mathcal{HI}$, but containing disjoint cycles. This is a contradiction.

To prove the reverse implication, consider a realistic string u having at least one positive pointer and avoiding disjoint cycles. Consider the set S of positive pointers of u, having the maximum number of overlapping negative pointers, and let S' be that subset of S consisting of pointers with the minimum number of overlapping positive pointers. Let $p \in S'$ be the pointer occurring first in u.

If $\mathbf{hi}_p(u)$ has a negative substring, then, as in the proof of Lemma 7, u is of the form $u = u_1 p q_1 \ldots q_i u_2 \overline{p}\, \overline{q}_1 \ldots \overline{q}_i u_3$, for some strings u_1, u_2, u_3 and pointers $q_1, \ldots, q_i$, $i \geq 1$. Note that in this case, u contains the disjoint cycle $\{pq_1, \overline{p}\,\overline{q}_1\}$, which is impossible. Thus, any elementary substring of $\mathbf{hi}_p(u)$ has at least one positive pointer.

Assume now that $v = \mathbf{hi}_p(u)$ contains a disjoint cycle P. The string u is realistic, i.e., $u = \pi(\delta)$, where δ is a realistic MDS descriptor. Since $\mathbf{hi}_p$ is applicable to u and $u = \pi(\delta)$, δ is either of the form $\delta = \delta_1(p,q)\delta_2(\overline{p},\overline{r})\delta_3$, or of the form $\delta = \delta_1(q,p)\delta_2(\overline{r},\overline{p})\delta_3$, for some pointers q, r and MDS sequences $\delta_1, \delta_2, \delta_3$. To make a choice, consider the former case, as the latter one can be solved similarly. Thus,

$$\delta = \delta_1(p,q)\delta_2(\overline{p},\overline{r})\delta_3 \quad \text{and} \quad \delta' = \mathbf{hi}_p(\delta) = \delta_1\overline{\delta}_2(\overline{q},\overline{r})\delta_3.$$

Since v contains a disjoint cycle, δ' has an IES cycle ι'; let $\mathcal{I}'$ be the set of IESs occurring in ι'. Since u contains no disjoint cycle, δ does not have IES cycles and so, $\mathcal{I}'$ must contain some IESs of δ' which do not occur in δ.

If δ_1, δ_2 are nonempty, let $\delta_i = \delta_i'(r_i, s_i)$, for $i \in \{1, 2\}$. Also, if $\delta_3 \neq \Lambda$, then let $\delta_3 = \delta_3'(r_3, s_3)$, otherwise let $s_3 = \overline{r}$.

We distinguish the following three main cases, depending on which of δ_1, δ_2 are empty MDS sequences.

(i) If both δ_1 and δ_2 are nonempty, then the only IES of δ', not occurring in δ is $(s_1, \overline{s}_2)$. Let $\mathcal{I}$ be the set formed by replacing $(s_1, \overline{s}_2)$ in $\mathcal{I}'$ with $(s_1, p), (s_2, \overline{p})$. The IESs in $\mathcal{I}$ form a cycle of δ, a contradiction.

(ii) If only one of δ_1, δ_2 is nonempty, then we have the following two subcases:

(ii.1) If $\delta_1 \neq \Lambda$, then the only IES of δ', not occurring in δ is $(s_1, \overline{q})$. Let $\mathcal{I}$ be the set formed by replacing $(s_1, \overline{q})$ in $\mathcal{I}'$ with $(s_1, p), (q, \overline{p})$. The IESs in $\mathcal{I}$ form a cycle of δ; a contradiction.

(ii.2) If $\delta_2 \neq \Lambda$, then the only IES of δ', not occurring in δ is $(\overline{s}_2, s_3)$ (i.e., the connector of δ'). Let $\mathcal{I}$ be the set formed by replacing $(\overline{s}_2, s_3)$ in $\mathcal{I}'$ with $(s_2, \overline{p}), (p, s_3)$. The IESs in $\mathcal{I}$ form a cycle of δ; a contradiction.

(iii) If $\delta_1 = \delta_2 = \Lambda$, then only IES of δ', not occurring in δ is $(\overline{q}, s_3)$ (i.e., the connector of δ'). Let $\mathcal{I}$ be the set formed by replacing $(\overline{q}, s_3)$ in $\mathcal{I}'$ with $(q, \overline{p}), (p, s_3)$. The IESs in $\mathcal{I}$ form a cycle of δ; a contradiction.

$\square$

The following theorem follows directly from Lemma 12.

Theorem 13. *A realistic word u has a successful reduction in $\mathcal{HI}$ if and only if for all legal substrings v of u, if $v = v_1 u_1 v_2 \ldots v_j u_j v_{j+1}$, where each u_i is a legal substring, then $v_1 v_2 \ldots v_{j+1}$ contains at least one positive pointer and avoids disjoint cycles.*

Thus, for instance, the pattern $M_1 M_4 \overline{M}_2 M_3$ has a successful reduction in $\mathcal{HI}$, while the patterns $M_1 \overline{M}_2 M_3$ and $\overline{M}_1 M_2 M_4 M_3$ have no such reductions in $\mathcal{HI}$.

5.7 $\{\mathbf{hi}, \mathbf{dlad}\}$-successful patterns

Using Theorems 11 and 13, we have the following result.

Theorem 14. *A realistic string u has a successful reduction in $\mathcal{HI} \cup \mathcal{DLAD}$ if and only if u avoids disjoint cycles.*

Theorem 14 has the following corollary.

Corollary 15. *An elementary realistic string u has a successful reduction in $\mathcal{HI} \cup \mathcal{DLAD}$ if and only if either u has a successful reduction in $\mathcal{HI}$, or u has a successful reduction in $\mathcal{DLAD}$.*

E.g., the pattern $\mu = M_1\overline{M}_2M_4M_3$, described by the elementary realistic string $u = 2\overline{3}\,\overline{2}434$ is successfully reducible in $\mathcal{HI} \cup \mathcal{DLAD}$: $\mathbf{dlad}_{3,4}(\mathbf{hi}_2(\mu)) = \Lambda$. By Corollary 15, it also has a successful reduction, either in $\mathcal{HI}$, or in $\mathcal{DLAD}$. Since u has positive pointers, $\mathcal{HI}$ must be the case. Indeed, $\mathbf{hi}_2(\mathbf{hi}_4(\mathbf{hi}_3(\mu))) = \Lambda$.

6 Discussion

In this paper we have investigated the IES/MDS patterns of micronuclear genes in ciliates. For each subset $\mathcal{S}$ of the set $\{ld, hi, dlad\}$ of molecular operations, we gave a characterization of all the patterns that can be assembled into a macronuclear gene using $\mathcal{S}$. We believe that these results provide a valuable insight into the micronuclear gene patterns, *and* into the nature of molecular operations that we consider. One can use our results to determine similarity of micronuclear genes and to classify these by their complexity. For example, two seemingly different micronuclear genes may be considered similar, if, e.g., both of them can be assembled using only *hi* operation. Also, the genes that need all three operations in any assembly strategy may be considered more complex than the genes needing only one or two of these operations. In this sense, see Section 5.4, the micronuclear *actin I* gene of *O.nova* is more complex than the micronuclear α-*TP* gene of *O.nova*. Also, one can order the three operations according to their complexity (e.g., $ld < hi < dlad$) and use this ordering to obtain a comparative classification of the complexity of *all* micronuclear genes.

It has been demonstrated in [12] that all known cases of gene assembly in ciliates can be accomplished using only the so called *simple* versions of *ld, hi*, and *dlad*. Therefore a logical next step is to characterize the micronuclear patterns that can be assembled by the simple versions of our operations.

Acknowledgments. Tero Harju, Ion Petre, and Grzegorz Rozenberg gratefully acknowledge support by TMR Network GETGRATS. The authors are indebted to David M. Prescott for his invaluable guidance in biology of ciliates.

References

[1] Ehrenfeucht, A., Harju, T., Petre, I., Prescott, D.M., Rozenberg, G., Formal systems for gene assembly in ciliates, to appear in Theoret. Comput. Sci. (2001).

[2] Ehrenfeucht, A., Petre, I., Prescott, D. M., and Rozenberg, G., Universal and simple operations for gene assembly in ciliates, in C.Martin-Vide, V.Mitrana (eds.) *Where Mathematics, Computer Science, Linguistics and Biology Meet*, Kluwer Academic Publishers, Dordrecht/Boston, 329–342 (2001).

[3] Ehrenfeucht, A., Petre, I., Prescott, D.M., Rozenberg, G., String and graph reduction systems for gene assembly in ciliates, to appear in Math. Struct. in Comput. Sci. (2001).

[4] Ehrenfeucht, A., Petre, I., Prescott, D.M., Rozenberg, G., Circular gene assembly in ciliates, manuscript (2001).

[5] Ehrenfeucht, A., Prescott, D.M., Rozenberg, G., Computational aspects of gene (un)scrambling in ciliates. In *Evolution as Computation*, L. Landweber, E. Winfree (eds.), 45–86, Springer Verlag, Berlin, Heidelberg (2001).

[6] Landweber, L.F. and Kari, L., The evolution of cellular computing: nature's solution to a computational problem. Proceedings of the 4th DIMACS meeting on DNA based computers, Philadelphia, 3–15 (1998).

[7] Landweber, L.F. and Kari, L., Universal molecular computation in ciliates. In *Evolution as Computation*, Landweber, L., Winfree, E. (eds.), Springer Verlag, Berlin, Heidelberg, (2001).

[8] Păun, G., Rozenberg, G., and Salomaa, A., *DNA Computing*, Springer-Verlag, Berlin, Heidelberg (1998).

[9] Prescott, D.M., Cutting, splicing, reordering, and elimination of DNA sequences in Hypotrichous ciliates. BioEssays, 14 (5): 317–324 (1992).

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

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

[12] Prescott, D.M., Ehrenfeucht, A., Rozenberg, G., Molecular operations for DNA processing in Hypotrichous ciliates, to appear in European Journal of Protistology (2001).