Formal systems for gene assembly in ciliates

Andrzej Ehrenfeuchta, Tero Harjub, Ion Petrec, a,b, *, David M. Prescottd,
Grzegorz Rozenbergc,a

a Department of Computer Science, University of Colorado, Boulder, CO 80309-0347, USA
b Department of Mathematics, University of Turku, FIN 20014, Turku, Finland
c Turku Centre for Computer Science, FIN 20520, Turku, Finland
d Department of Molecular, Cellular and Developmental Biology, University of Colorado, Boulder, CO 80309-0347, USA
e Leiden Institute of Advanced Computer Science, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, Netherlands

Abstract

DNA processing in ciliates, a very ancient group of organisms, is among the most sophisticated DNA processing in living organisms. It has a quite clear computational structure and even uses explicitly the linked list data structure! Particularly interesting from the computational point of view is the process of gene assembly from its micronuclear to its macronuclear form. We investigate here the string rewriting and the graph rewriting models of this process, involving three molecular operations, which together form a universal set of operations in the sense that they can assemble any macronuclear gene from its micronuclear form. In particular we prove that although the graph rewriting system is more “abstract” than the string rewriting system, no “essential information” is lost, in the sense that one can translate assembly strategies from one system into the other. © 2002 Elsevier Science B.V. All rights reserved.

Keywords: DNA computing; Ciliates; Gene assembly; Reduction systems; String rewriting; Graph rewriting; Overlap graphs

1. Introduction

We are now witnessing a big upsurge of research in bioinformatics, where computer science assists molecular biologists in understanding the structure and functioning of biomolecules, such as DNA and proteins, in living cells. Also, DNA computing is a fast growing interdisciplinary research area investigating the use of biomolecules for...
the purpose of computing (see, e.g., [11, 3]). Here molecular biology assists computer scientists to achieve a really bold goal: to replace (or to complement) the silicon-based computers by the DNA based computers. Most of research in DNA computing is concerned with the use of biomolecules in vitro, i.e., outside living cells (typically in test tubes). However, an important and exciting branch of DNA computing deals with the computational process in vivo, i.e., in living cells. Understanding the in vivo computational processes means the understanding of computational properties of biomolecules in their natural environment: the living cell. This area of research belongs also to bioinformatics, as it contributes essentially to the understanding of the computational nature of complex biological phenomena.

Ciliates, a very ancient group of organisms, have evolved one of the most intricate DNA processing in living organisms. They possess two kinds of nuclei: the micronucleus—essentially for storing DNA (until it is needed in the process of sexual reproduction) and the macronucleus—the “expression nucleus” which provides the RNA transcripts needed to operate the cell. When ciliates are starved, they proceed to sexual reproduction. It is in this process, after cell mating, that the micronucleus is “activated”: the micronuclear genome is converted into the macronuclear genome, where these two forms of the genome are drastically different. This conversion process is quite sophisticated and very interesting from the computational point of view.

The computational aspects of this genome conversion process have attracted the attention of the DNA computing community, see, e.g., [9, 10], and also [13], and the results obtained already reveal extraordinary features of DNA processing in ciliates, with some of them really stunning.

In this paper we continue our research into the formal aspects of DNA processing in ciliates. In [6] (a paper intended for biologists) we have introduced in a rather informal way two kinds of rewriting systems (one on strings and one on graphs) modeling the process of the macronuclear gene assembly from its micronuclear precursor. Then in [5] we have formalized these reduction systems and proved their equivalence from the point of view of possible strategies for gene assembly. The two models considered there were based on two molecular operations: ld-excision and hi-excision/reinsertion (or simply ld and hi). Then in [4] we have proved that by adding one more operation, dlad-excision/reinsertion (or simply dlad), one obtains a universal set of molecular operations in the sense that using these operations one can assemble the macronuclear gene from any micronuclear form (the previously considered two operations were not sufficient for universality).

In this paper we extend our formal string and graph reduction systems so that they also account for the new operation of dlad. We prove then that the extended systems preserve the equivalence property: string reduction and graph reduction are equivalent from the point of view of macrogene assembly strategies.

The paper is organized as follows.

In Section 2, we recall some basic features of the gene assembly in ciliates. We consider here the two operations from [5, 6]: ld and hi, as well as the operation of dlad from [12].
In Section 3, we formalize the structural information about the sequence of MDSs in the micronuclear, intermediate, or macronuclear form of a gene, through the notions of real MDS structures and realistic MDS descriptors. Then we formalize the molecular operations of \( ld, hi \) and \( dlad \) through formal operations on realistic MDS descriptors.

In Section 4, we extend the notion of the string pointer reduction system (from [5, 6]) so that it also takes into account the \( dlad \) operation. We prove then that from the point of view of successful reductions (strategies), these systems are equivalent to the reduction systems based on realistic MDS descriptors, considered in Section 2.

In Section 5, we extend the notion of the graph pointer reduction system (from [5, 6]) so that it also takes into account the \( dlad \) operation. We prove then that these systems are equivalent to string reduction systems as far as the successful strategies for gene assembly are concerned.

2. Gene assembly in ciliates

We will use the standard notation to denote single stranded and double stranded DNA: strings over the nucleotide alphabet \( \{A, C, G, T\} \) denote single stranded DNA, and double strings denote double stranded DNA (see, e.g., [11]). \textit{Perfect double strings} are double strings such that both the upper and the lower string are of the same length and each letter in any of the two strings has a corresponding letter (below or above it) in the other string. Thus, e.g., \( AACTGA \) \( TTGACT \) is a perfect double string, while \( TCAGTT \) \( TTGACT \) is not. Perfect double strings represent DNA molecules which are \textit{perfect duplexes}—in such a molecule each nucleotide has a complementary nucleotide in the other strand. The nonperfect double string above represents a DNA molecule with \textit{sticky ends} at both 5’-ends (see, e.g., [11]).

For a string \( \alpha \) (representing a single stranded molecule), its \textit{inversion} \( \overline{\alpha} \) is obtained by first taking the Watson–Crick complement of \( \alpha \) (i.e., replacing \( A \) by \( T \), \( C \) by \( G \), \( G \) by \( C \), and \( T \) by \( A \)), and then taking the mirror image of the resulting single string. Thus, e.g., for the string \( AACTGA \) its inversion is the string \( TCAGTT \). For a double string \( \gamma \) (representing a double stranded molecule), its \textit{inversion} \( \overline{\gamma} \) is obtained by first exchanging the two single strings of \( \gamma \) for each other, and then taking the mirror image of the resulting double string. Thus, e.g., for the (perfect) double string \( AACTGA \) \( TTGACT \) its inversion is the (perfect) double string \( TCAGTT \) \( AGTCAA \) while for the (nonperfect) double string \( TCAGTT \) \( TCAAGC \) its inversion is the (nonperfect) double string \( CGAACT \) \( TTGACT \).

A micronuclear gene \( \tau \) consists of a finite sequence of MDSs (macronuclear destined sequences), separated by IESs (internally eliminated sequences). The set \( \{M_1, \ldots, M_k\} \), \( k \geq 2 \), of MDSs occurring in \( \tau \) (each of them occurs only once) is such that, for
2 ≤ i ≤ k − 1, each $M_i$ has the structure

$$M_i = \left( \frac{p_i}{\tilde{p}_i}, \mu_i, \frac{p_{i+1}}{\tilde{p}_{i+1}} \right),$$

while $M_1$ and $M_k$ are of the form

$$M_1 = \left( \frac{p_2}{\tilde{p}_2}, \mu_1, p \right), \quad M_k = \left( \frac{p_2}{\tilde{p}_2}, \mu_k, e \right),$$

where $\mu_i$, $1 ≤ i ≤ k$, are double strands; we say that $\mu_i$ is the body of $M_i$. All of the double strands $\mu_i$ (and hence also $\tilde{\mu}_i$) are perfect double strands. All $p_i$, $2 ≤ i ≤ k$, are single strands and $\tilde{p}_i$ are their inversions. We refer to each $\frac{p_i}{\tilde{p}_i}$, $2 ≤ i ≤ k$, as a pointer. We say that the double strand $\frac{p_i}{\tilde{p}_i}$ is the incoming pointer of the MDS $M_i$, and the outgoing pointer of the MDS $M_{i-1}$, for all $2 ≤ i ≤ k$; thus, each pointer has two occurrences in $\tau$: one incoming and one outgoing. On the other hand, $b$ and $e$ are just symbolic markers designating the locations where an incipient macronuclear DNA molecule will be excised from macronuclear genome. We refer to them as the beginning and the ending marker, respectively. Thus in fact, each $M_i$, $2 ≤ i ≤ k − 1$, is a double stranded molecule of the form $\frac{p_i}{\tilde{p}_i} \mu_i \frac{p_{i+1}}{\tilde{p}_{i+1}}$ (resulting by “catenating” double strands $\frac{p_i}{\tilde{p}_i}, \mu_i, \frac{p_{i+1}}{\tilde{p}_{i+1}}$) $M_1$ is of the form $\mu_1 \frac{p_2}{\tilde{p}_2}$, and $M_k$ is of the form $\frac{p_k}{\tilde{p}_k} \mu_k$. Their inversions are double stranded molecules: $\tilde{M}_i$ of the form $\frac{p_{i+1}}{\tilde{p}_{i+1}} \tilde{\mu}_i \frac{p_i}{\tilde{p}_i}$, for $2 ≤ i ≤ k − 1$, $\tilde{M}_1$ of the form $\frac{p_2}{\tilde{p}_2} \tilde{\mu}_1$, and $\tilde{M}_k$ of the form $\frac{p_k}{\tilde{p}_k} \tilde{\mu}_k$.

It is important to notice here, that a pointer $\frac{p}{\tilde{p}}$ is always positioned at an end of an MDS, so typically at the boundary of an MDS and an IES. The same double sequence $\frac{p}{\tilde{p}}$ may be present somewhere else in $\tau$, but then this occurrence of $\frac{p}{\tilde{p}}$ is not a pointer.

The relationship between the micronuclear gene and the macronuclear gene is that the macronuclear gene is obtained by “gluing by overlapping” of $M_1, \ldots, M_k$ in this orthodox order (as illustrated in Fig. 1), while the sequence of MDSs forming the micronuclear gene (with interspersing IESs) is a permutation of the orthodox order with possibly some MDSs being inverted. For example, the micronuclear gene can be of the form $M_3 I_1 \tilde{M}_3 I_2 M_1 I_3 M_4 I_4 M_2$ (recall that $\tilde{M}_3$ is the inversion of $M_3$), where $I_1, \ldots, I_4$ are the interspersing IESs.

During the gene assembly process, which is the process of transformation of the micronuclear form of a gene into its macronuclear form, the IESs in the sequence are progressively excised, and the MDSs are spliced in the orthodox order, which is
suitable for transcription. It is postulated in [6, 12] that this transformation process is accomplished through the use of three molecular operations. As we explain in the sequel, the pointers are crucial for each of these operations. Thus, in order to keep the overall notation as simple as possible, we will simplify our notations for pointers by using positive integers $2, 3, \ldots, k - 1$, and $\bar{2}, \bar{3}, \ldots, \bar{k} - 1$ to denote them and their inversions (and we keep $b, e, \bar{b}, \bar{e}$ to denote the markers and their inversions). Then we set $y_{ENQ k}^{-1} = \{2, 3, \ldots, k - 1; \bar{2}, \bar{3}, \ldots, \bar{k} - 1\}$. As a matter of fact, we will refer to all elements of $y_{ENQ k}^{-1}$ as pointers, and for any pointer $p \in y_{ENQ k}^{-1}$ or marker $p \in \{b, e, \bar{b}, \bar{e}\}$, $\bar{p}$ denotes its inversion, where $\bar{p} = p$. For any $p \in \Pi_{k-1}$, we call \{p, $\bar{p}$\} the pointer set of $p$ and we denote it by $pts(p)$ (and by $pts(\bar{p})$). Thus using this simplified notation, $M_1 = (b, \mu_1, 2)$, $M_2 = (k, \mu_k, e)$, and $M_i = (i, \mu_i, i + 1)$, for all $2 \leq i \leq k - 1$. Then, for an MDS $M = (p, \mu, q)$, its inversion is $\bar{M} = (\bar{q}, \bar{\mu}, \bar{p})$.

Let $\tau$ be a micronuclear gene. We say that a pointer $p$ has an occurrence in $\tau$ if it is either the incoming, or the outgoing pointer of an MDS of $\tau$. The pointer $p$ has two adjacent occurrences in $\tau$ if it has two occurrences in $\tau$ separated by an IES only, or it has two occurrences at the ends of the molecule $\tau$, one at each end. Two pointers $p$ and $q$ are overlapping in $\tau$ if they both have two occurrences in $\tau$, with exactly one occurrence of $p$ between the two occurrences of $q$, or equivalently, with exactly one occurrence of $q$ between the two occurrences of $p$.

We say that $\tau$ has a direct repeat pattern $(p, p)$ of a pointer $p$ if $p$ has two occurrences in $\tau$. The molecule $\tau$ has an inverted repeat pattern $(p, \bar{p})$ of a pointer $p$ if both $p$ and its inverse $\bar{p}$ have one occurrence in $\tau$ and $p$ occurs before $\bar{p}$ (in $\tau$). We say that $\tau$ has an alternating direct repeat pattern of the pair of pointers $(p, q)$ if $p$ and $q$ are overlapping in $\tau$, and the first occurrence of $p$ comes before the first occurrence of $q$.

The three operations accomplishing the gene assembly are:
1. (loop, direct repeat)-excision (ld-excision, or just ld, for short),
2. (hairpin, inverted repeat)-excision/re insertion (hi-excision/reinsertion, or just hi, for short), and
3. (double loop, alternating direct repeat)-excision/reinsertion (dlad-excision/reinsertion, or just dlad, for short).

We now present briefly each of them, and refer for more details to [6, 12].

1. The operation of ld-excision is applicable to molecules having a direct repeat pattern $(p, p)$ of a pointer $p$. Such a molecule is folded into a loop aligned by the
direct repeat and then, the operation proceeds as shown in Fig. 2. The excision here involves staggered cuts (yielding sticky ends), and the operation yields two molecules: a linear one and a circular one. One of them consists of one IES only (“polluted” by one copy of the pointer \( p \)), while in the other molecule a bigger composite MDS is formed.

An application of the \( ld \) operation excises a circular molecule. Since none of our three operations joins two molecules together (in other words our model is the model of *intramolecular* processing), such an application may lead to a successful gene assembly only if either this circular molecule does not contain an MDS (it consists of an IES only), or it contains the whole gene in its micronuclear or intermediate form. The former case happens if the initial or the intermediate form of the macronuclear gene has two *adjacent* occurrences of the pointer to which \( ld \) is applied, i.e., two occurrences of the pointer separated by an IES. The latter happens if the initial or the intermediate form of the macronuclear gene begins with one occurrence of a pointer \( p \) and ends with the other occurrence of \( p \) and the \( ld \) operation is applied to \( p \) (such an application of \( ld \) is called *boundary*). However, once the two occurrences of \( p \) get into this boundary position, the application of any operation other than \( ld \) applied to \( p \) will not affect the position of the two copies of \( p \). Consequently, we will assume in the sequel that, unless it is otherwise clear, *if a boundary application of \( ld \) is used in a successful gene assembly, then it is the very last step of this assembly.*

2. The operation of *hi-excision/reinsertion* is applicable to molecules that have an inverted repeat pattern \( (p, \overline{p}) \) of a pointer \( p \). Such a molecule is folded into a hairpin aligned by the inverted repeat pair of pointers, and then the operation proceeds as shown in Fig. 3. The excision here involves staggered cuts; then through the reinsertion, the operation yields only one molecule. A bigger composite MDS is formed (as well as a bigger composite IES).
3. The operation of \textit{dlad-excision/reinsertion} is applicable to molecules that have an alternating direct repeat pattern of the pair of pointers \((p, q)\). Such a molecule is folded into a double loop with one loop aligned by \((p, p)\), and the other by \((q, q)\). Then, the operation proceeds as shown in Fig. 4. Once again, the excision here involves staggered cuts, but through reinsertion, the operation yields one molecule only.

3. MDS structures and descriptors

The process of gene assembly is accomplished through the use of the above three operations, which splice the MDSs in the orthodox order \(M_1, M_2, \ldots, M_k\) suitable for transcription. Hence, from the point of view of the gene assembly process, the structural information about the gene can be given by the sequence of MDSs only. Consequently, our first step towards the formalization of the assembly process in ciliates is to provide a representation for the MDS-sequence of a gene. We deal in this way with \textit{real MDS structures}, introduced in \([4, 5]\).

We use the finite alphabet \(\mathcal{M}_k = \{M_{i,j} \mid i, j \in \mathbb{N}, i \leq j \leq k\}\) to denote the MDSs of a given gene, where \(\mathbb{N}\) is the set of positive integers. The letters \(M_{i,i}\) may also be written as \(M_i\), and they are called \textit{elementary}, while the letters \(M_{i,j}, \text{ with } i < j\), are called \textit{composite}. A letter \(M_{i,j}\), with \(i < j\), is used to denote the composite MDS formed in the process of assembly by splicing the elementary MDSs \(M_i, M_{i+1}, \ldots, M_j\) through their pointers. Thus, the incoming pointer of \(M_{i,j}\) is \(i\) (or the marker \(b\), if \(i = 1\)), and its outgoing pointer is \(j + 1\) (or the marker \(e\), if \(j = k\)). We use the alphabet \(\mathcal{\hat{M}}_k = \{M_{i,j} \mid M_{i,j} \in \mathcal{M}_k\}\) to denote the inversions of the MDSs. Let \(\Theta_k = \mathcal{M}_k \cup \mathcal{\hat{M}}_k\).

A sequence \(M_{i_1,j_1}, M_{i_2,j_2}, \ldots, M_{i_n,j_n}\) is called \textit{orthodox} if \(i_1 = 1, i_l = 1 + j_{l-1}\), for all \(2 \leq l \leq n\), and \(j_n = k\). A sequence over \(\Theta_k\) is called a \textit{real MDS structure} if it is obtained by permuting an orthodox sequence, possibly inverting some of its elements.

The main idea behind our modeling of the assembly process is to keep the track of pointers (and markers) \textit{only}. Indeed, the pointers are used to align the fold required by each operation, and the assembly process continues as long as there are pointers left in the molecule, that is, as long as there are IESs left in the molecule (still separating MDSs). On the other hand, the macronuclear gene does not have IESs and hence, it
does not have pointers anymore. Hence, when the pointers “disappear”, the assembly process is completed.

In order to keep track of pointers only, we will denote each MDS $M = (p, \mu, q)$ by the ordered pair of its pointers/markers $(p, q)$, obtaining in this way MDS descriptors. More formally, we define the mapping $\psi$ on $\Theta_k$ by:

1. $\psi(M_{1,k}) = (b, e)$, and $\psi(M_{1,k}) = (\overline{e}, \overline{b})$,
2. $\psi(M_{1,i}) = (b, i + 1)$, and $\psi(M_{1,i}) = (\overline{i} + 1, \overline{b})$ for all $1 \leq i < k$,
3. $\psi(M_{i,k}) = (i, e)$, and $\psi(M_{i,k}) = (\overline{e}, \overline{i})$ for all $1 < i \leq k$,
4. $\psi(M_{i,j}) = (i, j + 1)$, and $\psi(M_{i,j}) = (\overline{j} + 1, \overline{i})$ for all $1 < i < j < k$,

where $b, e$ are reserved symbols. Then, for a sequence $X_1 \ldots X_l$ over $\Theta_k$,

$$\psi(X_1 \ldots X_l) = \psi(X_1) \ldots \psi(X_l).$$

Recall that the alphabet $\Pi_k = \{2, 3, \ldots, k, \overline{2}, \overline{3}, \ldots, \overline{k}\}$ denotes the pointers and their inverses. Also, let $\Psi = \{b, e, \overline{b}, \overline{e}\}$ be the set of markers, and $\Pi_{ex,k} = \Pi_k \cup \Psi$ be the extended alphabet. Let $I_k$ be the set of ordered pairs over $\Pi_{ex,k}$ consisting of:

- $(b, e)$, $(\overline{e}, \overline{b})$,
- $(b, i)$ $(\overline{i}, \overline{b})$ for all $2 \leq i \leq k - 1$,
- $(i, e)$ $(\overline{e}, \overline{i})$ for all $2 \leq i \leq k - 1$,
- $(i, j)$ $(\overline{j}, \overline{i})$ for all $2 \leq i < j \leq k - 1$.

A string over $I_k$ is called an MDS descriptor. Then, an MDS descriptor is realistic if $\mathrm{SO} = ((q, p) (p, q)) = ((\overline{q}, \overline{p}) (\overline{p}, \overline{q})$ for some real MDS structure $x$ (over $\Theta_k$).

Recall that for an alphabet $A$, the set of all strings over $A$ is denoted $A^*$, and $A$ denotes the empty string.

We can formalize now the molecular gene assembly operations through formal operations on realistic MDS descriptors. We will need the operation of reversed switch on $(I_k)^*$, defined as follows:

$$\text{rs}((p_1, q_1) \ldots (p_l, q_l)) = (\overline{q}_l, \overline{p}_l) \ldots (\overline{q}_1, \overline{p}_1)$$

for any $l \geq 0$. Corresponding to the three molecular operations $ld$, $hi$, and $dlad$, we will have three rules $\text{ld}$, $\text{hi}$, and $\text{dlad}$ which are defined on realistic MDS descriptors.

1. For each $p \in \Pi_k$, the $\text{ld}$-rule for $p$, denoted by $\text{ld}_p$, is defined as follows:

$$\text{ld}_p(\delta_1(q, p), p, r) = (\delta_1(q, p), r),$$

where $q, r \in \Pi_{ex,k}$, $\delta_1, \delta_2 \in (I_k)^*$, and $m_1, m_2 \in \Psi$.

The first case above corresponds to two adjacent occurrences of $p$ separated by an IES only, and the second case corresponds to two adjacent occurrences of $p$ at the ends of the molecule (the boundary application). This is illustrated in Fig. 5, where rectangles denote MDSs with their pointers indicated, the wiggly line denotes an arbitrary segment of a molecule (may contain both MDSs and IESs), and the straight line segment denotes an IES.
2. For each $p \in \Pi_k$, the hi-rule for $p$, denoted by $\text{hi}_p$, is defined as follows:

$$\text{hi}_p(\delta_1(p, q) \delta_2(\bar{p}, \bar{r}) \delta_3) = \delta_1 \text{rs}(\delta_2(\bar{q}, \bar{r}) \delta_3),$$

$$\text{hi}_p(\delta_1(q, p) \delta_2(\bar{r}, \bar{p}) \delta_3) = \delta_1(q, r) \text{rs}(\delta_2) \delta_3,$$

where $q, r \in \Pi_{ex,k}$ and $\delta_1, \delta_2 \in (I_k)^\ast$.

In the first case the first occurrence of $p$ is an incoming pointer and the second occurrence (in the “inverted” $(\bar{p}, \bar{r})$) is outgoing. In the second case, it is the other way around. This is illustrated in Fig. 6.

3. For each $p, q \in \Pi_k$, $p \neq q$, the dlad rule for $p$ and $q$, denoted as $\text{dlad}_{p,q}$, is defined as follows:

(a) $\text{dlad}_{p,q}(\delta_1(p, r_1) \delta_2(q, r_2) \delta_3(r_3, p) \delta_4(r_4, q) \delta_5) = \delta_1 \delta_4(r_4, r_2) \delta_3(r_3, r_1) \delta_2 \delta_5$,

(a.1) $\text{dlad}_{p,q}(\delta_1(p, r_1) \delta_2(q, r_2) \delta_4(r_4, q) \delta_5) = \delta_1 \delta_4(r_4, r_1) \delta_2 \delta_5$,

(b) $\text{dlad}_{p,q}(\delta_1(p, r_1) \delta_2(r_2, q) \delta_3(r_3, p) \delta_4(q, r_4) \delta_5) = \delta_1 \delta_4 \delta_3(r_3, r_1) \delta_2(r_2, r_4) \delta_5$,

(b.1) $\text{dlad}_{p,q}(\delta_1(p, q) \delta_3(r_3, p) \delta_4(q, r_4) \delta_5) = \delta_1 \delta_4 \delta_3(r_3, r_4) \delta_5$,

(c) $\text{dlad}_{p,q}(\delta_1(r_1, p) \delta_2(q, r_2) \delta_3(p, r_3) \delta_4(q, r_4) \delta_5) = \delta_1(r_1, r_3) \delta_4(r_4, r_2) \delta_3 \delta_2 \delta_5$,

(c.1) $\text{dlad}_{p,q}(\delta_1(r_1, p) \delta_2(q, r_2) \delta_3(p, q) \delta_5) = \delta_1(r_1, r_2) \delta_3 \delta_2 \delta_5$,

(d) $\text{dlad}_{p,q}(\delta_1(r_1, p) \delta_2(r_2, q) \delta_3(p, r_3) \delta_4(q, r_4) \delta_5) = \delta_1(r_1, r_3) \delta_4 \delta_3 \delta_2(r_2, r_4) \delta_5$,

where $r_1, r_2, r_3, r_4, r_5 \in \Pi_{ex,k}$, and $\delta_1, \delta_2, \delta_3, \delta_4, \delta_5 \in (I_k)^\ast$. In each of the above instances of $\text{dlad}_{p,q}$, the pointer $p$ overlaps with the pointer $q$, i.e., the molecule represented by the realistic MDS descriptor to which $\text{dlad}_{p,q}$ is applied, has a required double repeat pattern $(p, q, p, q)$.

In the above definition of $\text{dlad}$, we distinguish first the four main cases, illustrated in Fig. 7:

(a) the first occurrence of $p$ is incoming and the second outgoing, while the first occurrence of $q$ is incoming and the second outgoing;

(b) the two occurrences of $p$ are of the same sort as in (a), while the first occurrence of $q$ is outgoing and the second incoming;
(c) the first occurrence of $p$ is outgoing and the second incoming, while the two occurrences of $q$ are of the same sort as in (a);
(d) the two occurrences of $p$ are of the same sort as in (c), while the two occurrences of $q$ are of the same sort as in (b).

Then we have three “subcases”, illustrated in Fig. 8:
(a.1) the first occurrence of $q$ and the second occurrence of $p$ are the incoming and the outgoing pointers, respectively, of one MDS;
(b.1) the first occurrence of $p$ and the first occurrence of $q$ are the incoming and the outgoing pointer, respectively, of one MDS;
(c.1) the second occurrence of $p$ and the second occurrence of $q$ are the incoming and the outgoing pointer, respectively, of one MDS.

For a realistic MDS descriptor $\delta$ and rules $\rho_1, \ldots, \rho_l$, $l \geq 1$, from the set $\{\text{ld}_p, \text{hi}_p, \text{dlad}_{p,q}, \ | \ p,q \in \Pi_k\}$,

$$(\delta; \rho_1, \ldots, \rho_l)$$

is a reduction of $\delta$ if $\rho_1$ is applicable to $\delta$, and $\rho_i$ is applicable to $\rho_{i-1} \ldots \rho_1(\delta)$, for all $1 < i \leq l$. Such a reduction is successful if $\rho_1 \ldots \rho_l(\delta)$ is either $(b,e)$, or $(\bar{c},\bar{b})$. In this case, we also say that $(\delta; \rho_1, \ldots, \rho_l)$ is a successful strategy for $\delta$ (based on $\{\text{ld}, \text{hi}, \text{dlad}\}$).

**Example 1.** Let $\delta = (4, 5)(\bar{2},\bar{b})(5,e)(\bar{4},\bar{3})(\bar{3},\bar{2})$ be a realistic MDS descriptor. The rules $\text{ld}_3$, $\text{hi}_4$, and $\text{dlad}_{5,\bar{2}}$ are all applicable to $\delta$, and
Moreover, \((\delta; \text{dlad}_2, \text{hi}_4, \text{ld}_2)\) is a successful strategy for \(\delta\).

4. The string pointer reduction system

In Section 3, we have indicated how to model the gene assembly process by using realistic MDS descriptors. Using pointers, markers, and parentheses we were able to keep track of the (structure of) MDSs present in a molecule (at any moment in the process of gene assembly). In this section we will simplify our “instantaneous descriptions” by using legal strings which use pointers only. Then we will use rules which operate on strings of pointers only, obtaining in this way the string pointer reduction system \((\text{SPRS})\).

A legal string over \(\Pi_k^e\) is a string \(\pi \in \Pi_k^e\) such that for each \(p \in \Pi_k\), if \(\pi\) has one occurrence from \(\text{pts}(p)\), then \(\pi\) has exactly two occurrences from \(\text{pts}(p)\). Thus, either \(\pi\) contains one occurrence of \(p\) and one occurrence of \(\bar{p}\), in which case \(\text{pts}(p)\) is said to be positive in \(\pi\), or it contains two occurrences of \(p\), or two occurrences of \(\bar{p}\), in which case \(\text{pts}(p)\) is said to be negative in \(\pi\). A pointer \(p\) occurring in \(\pi\) is positive (negative, respectively) in \(\pi\).

For a realistic MDS descriptor \(\delta = (p_1, q_1) \ldots (p_m, q_m)\), by omitting parentheses, writing just the sequence of pointers, and deleting the markers, we obtain the legal string \(\pi_\delta\) corresponding to \(\delta\). We use \(\phi\) to denote this coding of \(\delta\) by \(\pi_\delta\) i.e., \(\phi(\delta) = \pi_\delta\).

For a string \(\pi = x_1x_2 \ldots x_n \in \Pi_k^e\), the reversed switch of \(\pi\), denoted \(\text{rs}(\pi)\), is the string \(\bar{x}_n \ldots \bar{x}_2 \bar{x}_1\). For a pointer \(p \in \Pi_k\) such that \(\{x_i, x_j\} \subseteq \text{pts}(p)\), for some positive integers \(i < j \leq n\), the p-interval of \(\pi\) is the substring \(x_i x_{i+1} \ldots x_j\). We say that two pointers \(p, q \in \Pi_k\) overlap in \(\pi\) if the p-interval of \(\pi\) overlaps with the q-interval of \(\pi\)—we also say then that \(\text{pts}(p)\) overlaps with \(\text{pts}(q)\) (recall that for \(i_2 > i_1\), the substrings \(x_{i_1} \ldots x_{j_1}\) and \(x_{i_2} \ldots x_{j_2}\) of \(\pi\) overlap iff \(i_2 \leq j_1\) and \(j_1 < j_2\)).

Example 2. If \(\delta\) is the realistic MDS descriptor \(\delta = (4, 5)(\tilde{2}, \tilde{b})(5, e)(3, 4)(\tilde{3}, \tilde{2})\), then its associated legal string is \(\pi_\delta = 45\tilde{2}534\tilde{3}\tilde{2}\). The 2-interval of \(\pi_\delta\) is the substring \(\tilde{2}534\tilde{3}\tilde{2}\), the 3-interval is \(34\tilde{3}\), the 4-interval is \(45\tilde{2}534\), and the 5-interval is \(525\). The pointer 3 overlaps in \(\pi_\delta\) with 4, but not with \(\tilde{2}\) or 5.

We introduce now the string pointer reduction system over \(\Pi_k\), a formal system abstracting the gene assembling process to the level of legal strings. In fact, \(\text{SPRS}\) defines reductions of legal strings over \(\Pi_k\) in such a way that each reduction removes all the occurrences from either one or two pointer sets.

Let \(\pi\) be a legal string over \(\Pi_k\) and let \(p, q \in \Pi_k\).
The string negative rule for $p$, denoted by $\text{snr}_p$, is applicable to $\pi$ if $pp$ is a substring of $\pi$, i.e., $\pi = \pi_1pp\pi_2$, for some strings $\pi_1, \pi_2$ over $\Pi_k$. The result of the application of $\text{snr}_p$ to $\pi$ is the legal string

$$\text{snr}_p(\pi) = \pi_1\pi_2.$$  

The string positive rule for $p$, denoted by $\text{spr}_p$, is applicable to $\pi$ if $p\overline{p}$ is a scattered substring of $\pi$, i.e., $\pi = \pi_1p\pi_2\overline{p}\pi_3$, for some strings $\pi_1, \pi_2, \pi_3$ over $\Pi_k$. The result of the application of $\text{spr}_p$ to $\pi$ is the legal string

$$\text{spr}_p(\pi) = \pi_1\pi_2.$$

The string double rule for $p$ and $q$, denoted by $\text{sdr}_{p,q}$, is applicable to $\pi$ if the pointers $p$ and $q$ overlap in $\pi$, with the first occurrence of $p$ preceding the first occurrence of $q$, i.e., $\pi = \pi_1p\pi_2q\pi_3p\pi_4q\pi_5$, for some strings $\pi_1, \pi_2, \pi_3, \pi_4, \pi_5$ over $\Pi_k$. The result of the application of $\text{sdr}_{p,q}$ to $\pi$ is the legal string

$$\text{sdr}_{p,q}(\pi) = \pi_1\pi_4\pi_3\pi_2\pi_5.$$  

Then the string pointer reduction system over $\Pi_k$ for the $\text{ld}$, $\text{hi}$, and $\text{dlad}$ operations ($\text{SPRS}_{\Pi_k}(\text{ld, hi, dlad})$) for short, or simply $\text{SPRS}$ if $\Pi_k$ is understood from the context of considerations and if it is understood that $\{\text{ld, hi, dlad}\}$ is the set of operations considered) is simply the set of the above operations. Thus, in this paper, $\text{SPRS} = \{\text{snr}_p, \text{spr}_p, \text{sdr}_{p,q} | p, q \in \Pi_k\}$.

For a legal string $\pi$ and a sequence of reduction rules $\rho_1, \ldots, \rho_l$, $l \geq 1$, from $\text{SPRS}$, $D = (\pi; \rho_1, \ldots, \rho_l)$ is a reduction (of $\pi$ by $\rho_1, \ldots, \rho_l$), if $\rho_1$ is applicable to $\pi$, and $\rho_i$ is applicable to $\rho_{i-1} \ldots \rho_1(\pi)$, for all $1 < i \leq l$. The result of $D$ is the legal string $\rho_1 \ldots \rho_l(\pi)$. We say that $D$ is successful if the result of $D$ is the empty string—we also say then that $\pi$ is successful (in $\text{SPRS}$).

**Example 3.** The rules $\text{hi}_2$ and $\text{dlad}_{4,5}$ are applicable to the legal string $\pi = 24\overline{236564}35$, and

$$\text{hi}_2(\pi) = 4\overline{3656435}, \quad \text{dlad}_{4,5}(\pi) = 236\overline{236}.$$  

The legal string $\pi$ is successful, and $(\pi; \text{dlad}_{4,5}, \text{hi}_6, \text{hi}_3, \text{ld}_2)$ is a successful reduction (of $\pi$ in $\text{SPRS}$).

For a legal string $\pi$, if either one of the rules $\text{snr}_p$ or $\text{spr}_p$ is applicable to $\pi$, for some $p \in \Pi_k$, then both occurrences in $\pi$ from $\text{pts}(p)$ are removed as the result of the reduction of $\pi$ by that rule. Also, if $\text{sdr}_{p,q}$ is applicable to $\pi$, for some $p, q \in \Pi_k$, then all occurrences in $\pi$ from $\text{pts}(p)$ and $\text{pts}(q)$ are removed as the result of the reduction of $\pi$ by $\text{sdr}_{p,q}$.

The following universality result was proved in [4].

**Theorem 1.** Every legal string $\pi$ over $\Pi_k$ is successful, i.e., there exists a sequence of operations $\rho_i \in \text{SPRS}$, $i = 1, 2, \ldots, m$, such that $\rho_m \ldots \rho_1(\pi) = \Lambda$. 
Hence, by this theorem, the set of the three operations \( ld, hi, \) and \( dlad \) is universal in the sense that a macronuclear gene can be assembled using these operations whatever its “scrambled” micronuclear version might be.

The effect of the gene assembly operations on the DNA molecules has been translated in the previous section into a formal system, transforming realistic MDS descriptors through the use of \( ld, hi \) and \( dlad \) operations. We will relate now this formal system to SPRS which operates on legal strings. In particular, we prove that if \( \delta \) is a realistic MDS descriptor, then every successful strategy for \( \delta \) (based on \( \{ld, hi, dlad\} \)) can be translated into a successful strategy for \( \pi_\delta \) (in SPRS). The following was proved in [5]:

**Theorem 2.** Let \( \delta \) be a realistic MDS descriptor, and let \( p \in \Pi_k \).

(i) The rule \( ld_p \) is applicable to \( \delta \) if and only if \( snr_p \) is applicable to \( \pi_\delta \). Moreover, in the case of applicability, if \( \pi' = snr_p(\pi_\delta) \) and \( \delta' = ld_p(\delta) \), then \( \pi' = \pi_{\delta'} \).

(ii) The rule \( hi_p \) is applicable to \( \delta \) if and only if \( spr_p \) is applicable to \( \pi_\delta \). Moreover, in the case of applicability, if \( \pi' = spr_p(\pi_\delta) \) and \( \delta' = hi_p(\delta) \), then \( \pi' = \pi_{\delta'} \).

We relate now the string reduction rule \( sdr \) to the operation \( dlad \) on realistic MDS descriptors.

**Lemma 3.** Let \( \delta \) be a realistic MDS descriptor and let \( p, q \in \Pi_k \) be two distinct pointers. If \( dlad_{p,q} \) is applicable to \( \delta \), then \( sdr_{p,q} \) is applicable to \( \pi_\delta \). Moreover, if \( \delta' = dlad_{p,q}(\delta) \), and \( \pi' = sdr_{p,q}(\pi_\delta) \), then \( \pi' = \pi_{\delta'} \).

**Proof.** If \( dlad_{p,q} \) is applicable to \( \delta \), then the pointer \( p \) overlaps with \( q \) in \( \delta \) and so, \( sdr_{p,q} \) is applicable to \( \pi_\delta \).

Applying \( dlad_{p,q} \) to \( \delta \) removes the occurrences of \( p \) and \( q \), and the sequence between the first occurrences of \( p \) and \( q \) in \( \delta \) is interchanged with the sequence between the second occurrences of \( p \) and \( q \) in \( \delta \). Consequently, \( \pi' = \pi_{\delta'} \).

**Lemma 4.** Let \( \delta \) be a realistic MDS descriptor and let \( p, q \in \Pi_k \) be two distinct pointers. If \( sdr_{p,q} \) is applicable to \( \pi_\delta \), then \( dlad_{p,q} \) is applicable to \( \delta \). Moreover, if \( \delta' = dlad_{p,q}(\delta) \), and \( \pi' = sdr_{p,q}(\pi_\delta) \), then \( \pi' = \pi_{\delta'} \).

**Proof.** If \( sdr_{p,q} \) is applicable to \( \pi_\delta \), then \( \pi_\delta = \pi_1 p \pi_2 q \pi_3 p \pi_4 q \pi_5 \), for some strings \( \pi_1, \pi_2, \pi_3, \pi_4, \pi_5 \). Thus, both \( p \) and \( q \) have two occurrences in \( \delta \), a left one and a right one. We consider the following cases:

1. The first occurrence of \( p \) in \( \delta \) is left, as well as the first occurrence of \( q \) in \( \delta \).
2. The first occurrence of \( p \) in \( \delta \) is right, as well as the first occurrence of \( q \) in \( \delta \).
3. The first occurrence of \( p \) in \( \delta \) is left, and the first occurrence of \( q \) in \( \delta \) is right.
4. The first occurrence of \( p \) in \( \delta \) is right, and the first occurrence of \( q \) in \( \delta \) is left.

We prove the claim only for the first case, as the reasoning for the other cases is similar. Assume then that the first occurrence of \( p \) in \( \delta \) is left, and the first occurrence
of $q$ is also left in $\pi$. Clearly, between the first occurrences of $p$ and $q$ in $\pi$, there must be at least one pointer or marker: $\delta = \delta_1(p, r_1)\delta_2(q, r_2)\delta'$, with $r_1, r_2 \in \Pi_{\mathbb{N}, k}$, and $\delta_1, \delta_2, \delta'$ strings over $I_k$. Now, if the string $\pi_3$ between the first occurrence of $q$ and the second occurrence of $p$ in $\pi_5$ is nonempty, then necessarily, there must be at least one pointer or marker between the similar occurrences of the pointers $p$ and $q$ in $\pi_5$: $\pi_5 = \pi_6(p; r_1)\pi_7(q; r_2)\pi_8(r_3; p)\pi_9(r_4; q)\pi_5'$, for some $r_3, r_4 \in \Pi_{\mathbb{N}, k}$, and $\delta_3, \delta_4, \delta_5$ strings over $I_k$. On the other hand, if $\pi_3$ is the empty string, then either between the first occurrence of $q$ and the second occurrence of $p$ there is a marker, which is a particular case of the previous one, or there is no pointer or marker between them: $\delta = \delta_1(p, r_1)\delta_2(q, r_2)\delta_3(r_3; p)\delta_4(r_4; q)\delta_5$. In both cases, $\text{dlad}_{p, q}$ is applicable to $\delta$: in the former case by the instance (a) in the definition of $\text{dlad}_{p, q}$, and in the latter case by the instance (a.1) in the same definition.

The second part of the claim follows from Lemma 3.

From Lemmata 3 and 4, we obtain the intertranslatability between the operation $\text{dlad}$ on realistic MDS descriptors and the reduction rule $\text{sdr}$ on legal strings.

**Corollary 5.** Let $\delta$ be a realistic MDS descriptor, and let $p, q \in \Pi_{\mathbb{N}, k}$ be two distinct pointers. The rule $\text{dlad}_{p, q}$ is applicable to $\delta$ if and only if $\text{sdr}_{p, q}$ is applicable to $\pi_5$. Moreover, in the case of applicability, if $\pi' = \text{sdr}_{p, q}(\pi_5)$ and $\delta' = \text{dlad}_{p, q}(\delta)$, then $\pi' = \pi_5'$.

This result, combined with Theorem 2 demonstrates the intertranslatability between the successful strategies for realistic MDS descriptors, and the successful reductions of legal strings.

### 5. The graph pointer reduction system

In this section, we will use graphs to represent the structure of pointers in legal strings—a legal string $\pi$ will be translated into a graph $\gamma_\pi$, where a pointer set is represented by a node. Then we will consider the graph pointer reduction system (GPRS) operating on such graphs—here a single reduction will correspond to the removal of either one node (when modeling $\text{ld}$ or $\text{hi}$) or two nodes (when modeling $\text{dlad}$).

A graph $\gamma = (V, E)$ consists of a set $V$ of nodes together with a set $E$ of undirected edges, where $E \subseteq \{(x, y) \mid x, y \in V, x \neq y\}$. Note that we do not allow multiple edges between two nodes, and we do not allow loops $\{x, x\}$ on the nodes. For a graph $\gamma$, we shall also write $V_\gamma$ and $E_\gamma$ for the set of nodes and the set of edges of $\gamma$, respectively.

If in a graph $\gamma$ we have $\{x, y\} \in E_\gamma$, then the nodes $x$ and $y$ are adjacent to each other. For a node $x \in V_\gamma$, let

$$N_\gamma(x) = \{y \mid \{x, y\} \in E_\gamma\}$$
Fig. 9. An example of a signed graph.

Fig. 10. The graphs obtained by applying: (a) $\text{gir}_4$, (b) $\text{gpr}_4$, and (c) $\text{gdr}_{4.5}$ to the graph given in Fig. 9.

be the neighborhood of $x$ in $\gamma$. We let

$$\hat{N}_\gamma(x) = N_\gamma(x) \cup \{x\}$$

be the closed neighborhood of the node $x$. A node $x$ is said to be isolated, if $N_\gamma(x) = \emptyset$.

A signed graph $\gamma = (V, E, \theta)$ consists of a graph $(V, E)$ and a sign function $\theta: V \to \{+, -\}$ of the nodes. A node $p$ is said to be positive (negative) in $\gamma$, if $\theta(p) = +$ ($\theta(p) = -$, respectively).

Let $S \subseteq V$ be a subset of the nodes of a signed graph $\gamma = (V, E, \theta)$. The graph $\gamma' = (V, E', \theta')$ is obtained from $\gamma$ by complementing $S$, if for all pairs $\{x, y\}$ of $V$ with $x \neq y$, we have $\{x, y\} \in E'$ if and only if

- $x \in S$ or $y \in S$ and $\{x, y\} \in E$,
- $x \in S$ and $y \in S$ and $\{x, y\} \notin E$,

and $\theta'(p) = -\theta(p)$ for all $p \in S$, and $\theta'(p) = \theta(p)$ if $p \notin S$. Moreover, if for a node $p \in V$ we complement $N_\gamma(p)$, then we get the local complementation at $p$ (in $\gamma$), denoted by $\text{gir}_p(\gamma)$.

Example 4. The graph $\gamma$ given in Fig. 9 is a signed graph and $N_\gamma(4) = \{2, 3, 5\}$. The graph $\text{gir}_4(\gamma)$ is given in Fig. 10(a).

Let $\Delta$ be an alphabet. A string $\delta$ over $\Delta$ is called a double occurrence string, if each letter of $\Delta$ either occurs in $\delta$ exactly twice, or it does not occur at all. The (unsigned) overlap graph of a double occurrence string $\delta$ is the graph $\gamma_\delta = (\Delta_k, E)$, where

$$E = \{\{p, q\} \mid p \text{ and } q \text{ overlap in } \delta\}.$$
presentation of these graphs is obtained by intersections of chords of a circle, see again [8]. For this reason the interval overlap graphs are often called circle graphs. We refer to [2, 7] for characterizations of these graphs.

Let \( A_k = \{2, 3, \ldots, k\} \). For a legal string \( \pi \in \Pi_k^* \), we let \( \|\pi\| \) denote the double occurrence string resulting from \( \pi \) by removing “bars”, i.e., we identify \( p \) and \( \bar{p} \) with the corresponding letter of \( A_k \), for all letters \( p \in \Pi_k \).

Let \( \pi \) be a legal string over \( \Pi_k \). The signed overlap graph of \( \pi \) is the signed graph \( \gamma_\pi = (A_k, E, \theta) \), where \( (A_k, E) \) is the overlap graph of the double occurrence string \( \|\pi\| \), and the sign function \( \theta: A_k \to \{+, -\} \) is defined by

\[
\theta(p) = + \iff \text{the letter } p \text{ is positive in } \pi.
\]

Therefore \( \theta(p) = + \), if \( pp \bar{p} \) or \( \bar{p}p \) is a scattered substring of \( \pi \) and \( \theta(p) = - \) if \( pp \) or \( \bar{p}p \) is a scattered substring of \( \pi \) for the letters \( p \in A_k \) that occur in \( \pi \).

Example 5. For the legal string \( \pi = 24\bar{2}3\bar{6}\bar{5}6435 \), its signed overlap graph is given in Fig. 9.

The mapping \( \pi \mapsto \gamma_\pi \) of the legal strings to the signed overlap graphs is many-to-one. E.g., for the legal strings \( \pi_1 = ppqq \) and \( \pi_2 = pqqp \), \( \gamma_{\pi_1} = \gamma_{\pi_2} \).

Let \( \pi \) be a legal string over \( \Pi_k \), \( \gamma_\pi \) its signed overlap graph, and \( p, q \in \Pi_k \).

The graph negative rule for \( p \), denoted by \( \text{gnr}_p \), is applicable to \( \gamma_\pi \) if the node \( p \) is isolated and negative in \( \gamma_\pi \). The result of the application of \( \text{gnr}_p \) to \( \gamma_\pi \) is the signed overlap graph \( \text{gnr}_p(\gamma_\pi) \), obtained from \( \gamma_\pi \) by removing the node \( p \).

The graph positive rule for \( p \), denoted by \( \text{gpr}_p \), is applicable to \( \gamma_\pi \) if the node \( p \) is positive in \( \gamma_\pi \). The result of the application of \( \text{gpr}_p \) to \( \gamma_\pi \) is the signed overlap graph \( \text{gpr}_p(\gamma_\pi) \), obtained from \( \text{gir}_p(\gamma_\pi) \) by removing the node \( p \) (recall that \( \text{gir}_p(\gamma_\pi) \) is the local complementation of \( \gamma_\pi \) at \( p \)).

Example 6. The rule \( \text{gpr}_4 \) is applicable to the graph \( \gamma \) given in Fig. 9 and the result, \( \text{gpr}_4(\gamma) \), is given in Fig. 10(b).

The graph double rule for \( p \) and \( q \), denoted by \( \text{gdr}_{p,q} \), is applicable to \( \gamma_\pi \) if the nodes \( p \) and \( q \) are negative and adjacent in \( \gamma_\pi \). If \( \gamma_\pi = (V, E, \theta) \), then the result of the application of \( \text{gdr}_{p,q} \) to \( \gamma_\pi \) is the signed overlap graph \( \text{gdr}_{p,q}(\gamma_\pi) = (V \setminus \{p,q\}, E', \theta') \) obtained as follows: \( \theta' \) equals \( \theta \) restricted to \( V \setminus \{p,q\} \), and \( E' \) is obtained from \( E \) by complementing the edges between the sets \( N_{\gamma_\pi}(p) \) and \( N_{\gamma_\pi}(q) \). This means that the status of a pair \( \{x,y\} \) (for \( x,y \in V \setminus \{p,q\} \)) as an edge will change if and only if

- \( x \in N_{\gamma_\pi}(p) \setminus N_{\gamma_\pi}(q) \) and \( y \in N_{\gamma_\pi}(q) \);
- \( x \in N_{\gamma_\pi}(p) \cap N_{\gamma_\pi}(q) \) and \( y \in N_{\gamma_\pi}(q) + N_{\gamma_\pi}(p) \);
- \( x \in N_{\gamma_\pi}(q) \setminus N_{\gamma_\pi}(p) \) and \( y \in N_{\gamma_\pi}(p) \),

where \( + \) denotes the symmetric difference of the two neighborhoods (i.e., \( N_{\gamma_\pi}(q) \setminus N_{\gamma_\pi}(p) \cup N_{\gamma_\pi}(p) \setminus N_{\gamma_\pi}(q) \)).
Example 7. Let $\gamma$ be the signed overlap graph given in Fig. 9. Then $\text{gdr}_{4,5}$ is applicable to $\gamma$,

$$N_\gamma(4) = \{2, 3, 5\}, \quad N_\gamma(5) = \{3, 4, 6\},$$

and $\text{gdr}_{4,5}(\gamma)$ is the graph given in Fig. 10(c).

The graph pointer reduction system over $\Pi_k$ for the $\text{ld}$, $\text{hi}$, and $\text{dlad}$ operations ($\text{GPRS}_{\Pi_k}(\text{ld, hi, dlad})$) for short, or simply $\text{GPRS}$ if $\Pi_k$ is understood from the context of considerations and if it is understood that $\{\text{ld, hi, dlad}\}$ is the set of operations considered) is simply the set of the above reduction rules. Thus, $\text{GPRS} = \{\text{gnr}_p, \text{gpr}_p, \text{gdr}_p, q | p, q \in \Pi_k\}$.

Let $\pi$ be a legal string and $\gamma_\pi$ its signed overlap graph. For a sequence of reduction rules $\rho_1, \ldots, \rho_l$, $l \geq 1$, from $\text{GPRS}$, $D = (\gamma_\pi; \rho_1, \ldots, \rho_l)$ is a reduction (of $\gamma_\pi$ by $\rho_1, \ldots, \rho_l$), if $\rho_1$ is applicable to $\gamma_\pi$, and $\rho_i$ is applicable to $\rho_{i-1} \ldots \rho_1(\gamma_\pi)$, for all $1 < i \leq l$. The result of $D$ is the signed overlap graph $\rho_1 \ldots \rho_l(\gamma_\pi)$. We say that $D$ is successful if the result of $D$ is the empty graph—we also say then that $\gamma_\pi$ is successful (in $\text{GPRS}$).

Example 8. The signed overlap graph $\gamma_\pi$ given in Fig. 9 is successful and $(\gamma_\pi; \text{gdr}_{4,5}, \text{gpr}_2, \text{gpr}_3, \text{gnr}_6)$ is a successful reduction.

We study now the relation between the string pointer reduction system $\text{SPRS}$ and the graph pointer reduction system $\text{GPRS}$.

The following result is proved in [5].

**Lemma 6.** Let $\pi$ be a legal string and $p \in \Pi$. If $\text{snr}_p$ is applicable to $\pi$, then $\text{gnr}_p$ is applicable to $\gamma_\pi$, and

$$\gamma_{\text{snr}_p(\pi)} = \text{gnr}_p(\gamma_\pi).$$

For a pointer $p \in \Pi_k$, the auxiliary operation $\text{snr}_p$ is defined on the legal strings of the form $\pi = \pi_1 p \pi_2 \pi_1 \pi_3, \pi_1, \pi_2, \pi_3 \in \Pi_k^*$, $p_1 \in \text{pts}(p)$, and the result is defined by:

$$\text{snr}_p(\pi) = \pi_1 p \pi_2(\pi_3).$$

Hence, applying $\text{snr}_p$ to $\pi$ yields a legal string obtained from $\pi$ by inverting the “open $p$-interval” $\pi_2$.

The following was proved in [5] for signed overlap graphs of legal strings. It is also considered in [1] for the circle graphs of double occurrence strings.

**Lemma 7.** Let $\pi$ be a legal string and $p \in \Pi$. If $\text{snr}_p$ is applicable to $\pi$, then $\text{gnr}_p$ is applicable to $\gamma_\pi$, and

$$\gamma_{\text{snr}_p(\pi)} = \text{gnr}_p(\gamma_\pi).$$
Moreover, if $\mathsf{spr}_p$ is applicable to $\pi$, then $\mathsf{gpr}_x$ is applicable to $\gamma_{\pi}$, and

$$\gamma_{\mathsf{spr}_p(\pi)} = \mathsf{gpr}_p(\gamma_{\pi}).$$

**Proof.** Let $\pi = \pi_1 p\pi_2 p\pi_3$, with $\pi_1, \pi_2, \pi_3 \in \Pi_k^*$, $p_1 \in \mathsf{pts}(p)$. Hence $\mathsf{sir}_p(\pi) = \pi_1 p \, \mathsf{rs}(\pi_2) p_1 \pi_3$. The status of a pair $\{x, y\}$ as an edge is changed in the transformation $\gamma_{\pi} \mapsto \gamma_{\mathsf{sir}_p(\pi)}$ if and only if exactly one of the occurrences of both $x$ and $y$ is in $\pi_2$. This is equivalent to $x, y \in N_{\gamma_{\pi}}(p)$. Also, it is clear that the sign of a node $x$ changes if and only if $x$ has exactly one occurrence in $\pi_2$, or equivalently if $x \in N_{\gamma_{\pi}}(p)$. Hence $\gamma_{\mathsf{sir}_p(\pi)} = \mathsf{gir}_p(\gamma_{\pi})$. The second claim follows immediately from the first one. \qed

Our proof of the following lemma is adapted from [1], where a closely related result is stated for (unsigned) overlap graphs and double occurrence strings.

**Lemma 8.** Let $\pi$ be a legal string and $p, q \in \Pi_k$. If $\mathsf{sdr}_{p, q}$ is applicable to $\pi$, then $\mathsf{gdr}_{p, q}$ is applicable to $\gamma_{\pi}$. Moreover,

$$\gamma_{\mathsf{sdr}_{p, q}(\pi)} = \mathsf{gdr}_{p, q}(\gamma_{\pi}).$$

**Proof.** Note first that the nodes $p$ and $q$ are negative in $\gamma_{\pi}$, and thus the graph $\mathsf{gdr}_{p, q}(\gamma_{\pi})$ is well defined.

We shall decompose the operation $\mathsf{sdr}$ to three applications of local complementation:

$$\mathsf{sdr}_{p, q} = \mathsf{spr}_p \circ \mathsf{spr}_q \circ \mathsf{sir}_p.$$

To see this, let $\pi = \pi_1 p\pi_2 q\pi_3 p\pi_4 q\pi_5$ for the substrings $\pi_i$, $i = 1, 2, \ldots, 5$. Then

$$\mathsf{spr}_p \circ \mathsf{spr}_q \circ \mathsf{sir}_p(\pi) = \mathsf{spr}_p \circ \mathsf{spr}_q(\pi_1 p \, \mathsf{rs}(\pi_3) q \, \mathsf{rs}(\pi_2) p\pi_4 q\pi_5)$$

$$= \mathsf{spr}_p(\pi_1 p \, \mathsf{rs}(\pi_3) \mathsf{rs}(\pi_4) \mathsf{rs}(\pi_5))$$

$$= \pi_1 \pi_4 \pi_3 \pi_5 = \mathsf{sdr}_{p, q}(\gamma_{\pi}),$$

where in the first line $q$ and in the second line $p$ become positive, and therefore the operations $\mathsf{spr}_q$ and $\mathsf{spr}_p$ do apply there. It is clear that the positive/negative character of the letters does not change in the transformation $\pi \mapsto \mathsf{sdr}_{p, q}(\pi)$.

Consider the transformation $\gamma_{\pi} \mapsto \gamma_{\mathsf{sdr}_{p, q}(\pi)}$. We will split it into three steps, corresponding to the splitting of $\mathsf{sdr}_{p, q}$ into three steps above: $\gamma_{\pi} \mapsto \gamma_{\mathsf{sir}_p(\pi)}$, $\gamma_{\mathsf{sir}_p(\pi)} \mapsto \gamma_{\mathsf{spr}_q(\mathsf{sir}_p(\pi))}$, and $\gamma_{\mathsf{spr}_q(\mathsf{sir}_p(\pi))} \mapsto \gamma_{\mathsf{sdr}_{p, q}(\pi)}$.

In the first step: $\gamma_{\pi} \mapsto \gamma_{\mathsf{sir}_p(\pi)}$, the set $N_{\gamma_{\pi}}(p)$ is complemented, and since $q \in N_{\gamma_{\pi}}(p)$, $q$ becomes positive in $\gamma_{\mathsf{sir}_p(\pi)}$, and

$$N_{\gamma_{\mathsf{sir}_p(\pi)}}(q) = N_{\gamma_{\pi}}(p) + \hat{N}_{\gamma_{\pi}}(q).$$

In the second step, $\gamma_{\mathsf{sir}_p(\pi)} \mapsto \gamma_{\mathsf{spr}_q(\mathsf{sir}_p(\pi))}$, the set $N_{\gamma_{\mathsf{sir}_p(\pi)}}(q)$ is complemented, and $q$ is removed from the graph. Since $p \in N_{\gamma_{\mathsf{sir}_p(\pi)}}(q)$, $p$ becomes positive in $\gamma_{\mathsf{spr}_q(\mathsf{sir}_p(\pi))}$, and

$$N_{\gamma_{\mathsf{spr}_q(\mathsf{sir}_p(\pi))}}(p) = N_{\gamma_{\pi}}(q) \{p\}.$$
Finally, in the third step, \( \gamma_{\text{sp}}(\text{sr}(\pi)) \mapsto \gamma_{\text{sd}}(\pi) \), the set \( N_{\gamma_{\text{sp}}(\text{sr}(\pi))}(p) \) is complemented and the node \( p \) is removed.

The pairs inside the three sets \( N_{\gamma_{\text{sp}}(\pi)}(p) \setminus N_{\gamma_{\text{sp}}(\pi)}(q), N_{\gamma_{\text{sp}}(\pi)}(p) \cap N_{\gamma_{\text{sp}}(\pi)}(q), \) and \( N_{\gamma_{\text{sp}}(\pi)}(q) \setminus N_{\gamma_{\text{sp}}(\pi)}(p) \) are complemented twice in the above transformations, and hence the status of these pairs as edges remains the same in \( \gamma_{\text{sd}}(\pi) \) as in \( \gamma_{\text{sp}}(\pi) \). On the other hand, the pairs between the above three sets are complemented once in the transformation, and therefore they are complemented in \( \gamma_{\text{sp}}(\pi) \mapsto \gamma_{\text{sd}}(\pi) \). This proves the claim.

The following result follows from Lemmata 6, 7, and 8:

**Theorem 9.** Every string reduction \( D = (\pi; \rho_1, \ldots, \rho_l) \) in SPRS translates into a graph reduction \( D' = (\gamma_{\text{sp}}; \rho'_1, \ldots, \rho'_l) \) in GPRS, by translating:
- each SPRS reduction rule \( \text{sn}(p) \) into the GPRS reduction rule \( \text{gr}(p) \);
- each SPRS reduction rule \( \text{sp}(p) \) into the GPRS reduction rule \( \text{gp}(p) \);
- each SPRS reduction rule \( \text{sd}(p, q) \) into the GPRS reduction rule \( \text{gd}(p, q) \).

Consequently, if \( D \) is successful, then so is \( D' \). Thus, for every legal string \( \pi, \gamma_{\text{sp}} \) is a successful signed overlap graph.

We move now to prove the reverse implication, namely that any successful reduction in the graph pointer reduction system has an equivalent successful reduction in the string pointer reduction system.

We will need the following result from [5].

**Lemma 10.** Let \( \pi \) be a legal string and let \( p \in \Pi_k \). If \( \text{gp}(p) \) is applicable to \( \gamma_{\text{sp}} \), then \( \text{sp}(p) \) is applicable to \( \pi \). Moreover, if \( \pi' = \text{sp}(\pi) \), and \( \gamma' = \text{gp}(\gamma_{\text{sp}}) \), then \( \gamma_{\text{sp}} = \gamma' \).

An analogous result holds also for the double rules:

**Lemma 11.** Let \( \pi \) be a legal string and let \( p, q \in \Pi_k \). If \( \text{gd}(p, q) \) is applicable to \( \gamma_{\text{sp}} \), then \( \text{sd}(p, q) \) is applicable to \( \pi \). Moreover, if \( \pi' = \text{sd}(\pi) \), and \( \gamma' = \text{gd}(\gamma_{\text{sp}}) \), then \( \gamma_{\text{sp}} = \gamma' \).

**Proof.** If \( \text{gd}(p, q) \) is applicable to \( \gamma_{\text{sp}} \), then the vertices \( p \) and \( q \) are negative and adjacent in \( \gamma_{\text{sp}} \). Consequently, the pointers \( p \) and \( q \) are negative and overlapping in \( \pi \) and thus, \( \text{sd}(p, q) \) is applicable to \( \pi \). Moreover, by Lemma 8, \( \gamma_{\text{sp}} = \gamma' \).

There is no similar result for the negative rules. Indeed, for the legal string \( \pi = pqqp \), the node \( p \) is negative and isolated in \( \gamma_{\text{sp}} \) and thus, it can be reduced by applying \( \text{gn}(p) \) to \( \gamma_{\text{sp}} \), but on the other hand, \( \text{sn}(p) \) is not applicable to \( \pi \). However, we will show that one can translate successful reductions in GPRS into successful string reductions in SPRS. For this, we will need the following result proved in [5], and the notion of permuting a reduction.
For two GPRS reductions \(D=(\gamma; \rho_1, \ldots, \rho_l)\) and \(D'=(\gamma'; \rho'_1, \ldots, \rho'_l)\), we say that \(D'\) is a permutation of \(D\) if the sequence of rules \(\rho'_1, \ldots, \rho'_l\) is a permutation of the sequence \(\rho_1, \ldots, \rho_l\). The following result is from [5]:

**Lemma 12.** Let \(\pi\) be a legal string. If \(D'=(\gamma; gnr_p, \ldots, gnr_q)\) is a successful reduction in GPRS, then there exists a successful reduction \(D=(\pi; snr_{q'}, \ldots, snr_{p'})\) of \(\pi\) in SPRS, such that \(D''=(\gamma; gnr_{q'}, \ldots, gnr_{q'})\) is a successful reduction in GPRS and \(D''\) is a permutation of \(D\).

**Theorem 13.** Let \(\pi\) be a legal string. Then, for every successful graph reduction \(D'=(\gamma; \rho_1, \ldots, \rho_l)\) of \(\gamma\) in GPRS, there exists a permutation \(D=(\pi; \rho'_1, \ldots, \rho'_l)\) in SPRS, by translating

- each GPRS reduction rule \(gnr_p\) into the SPRS reduction rule \(snr_p\);
- each GPRS reduction rule \(gpr_p\) into the SPRS reduction rule \(spr_p\);
- each GPRS reduction rule \(gdr_p,q\) into the SPRS reduction rule \(sdr_p,q\).

**Proof.** Let \(\pi\) be a legal string and let \(D'\) be a successful reduction of \(\gamma\) in GPRS (such a \(D'\) exists by Theorem 9). Since applying graph negative rules amounts to removing isolated nodes, such rules can be grouped together as the “tail” of a given reduction (yielding a so called canonical reduction, see [5]). In this way we transform \(D'\) to a successful canonical reduction \(D_1=(\gamma; \rho_1, \ldots, \rho_l, gnr_{q'}, \ldots, gnr_{q'})\), for some vertices \(q_1, \ldots, q_i\), and some positive or double rules \(\rho_1, \ldots, \rho_i\). By Lemmata 10 and 11,

\[
(\pi; \rho'_1, \ldots, \rho'_l)
\]

is a reduction in SPRS, where \(\rho'_k\) is obtained from \(\rho_k\), \(1 \leq k \leq i\) in the following way:

- if \(\rho_k = gpr_p\), for some \(p \in \Pi_k\), then \(\rho'_k = spr_p\); and if \(\rho_k = gdr_p,q\), for some \(p,q \in \Pi_k\), then \(\rho'_k = sdr_p,q\). Moreover, for

\[
\gamma' = \rho_1 \ldots \rho_l(\gamma)\quad \text{and} \quad \pi' = \rho'_1 \ldots \rho'_l(\pi),
\]

we have \(\gamma' = \gamma'\). Then, by Lemma 12, there exists a successful string reduction \((\pi'; snr_{q'}, \ldots, snr_{q'})\) in SPRS, proving the theorem. \(\square\)

6. Discussion

We have considered in this paper three molecular operations: \(ld, hi,\) and \(dlad\), and their formalizations in three different formal systems: MDS descriptor reduction systems, string pointer reduction systems, and graph pointer reduction systems (in this order of increasing abstraction). We have then established the intertranslatability, in a rather strong sense, between (the reductions in) these formal systems. We believe that this research together with the work on intermolecular framework (see, e.g., [9, 10])
increases our understanding of DNA computing *in vivo* actually taking place in living organisms.

This paper completes (together with [4, 5]), the setting up of a formal framework for (intramolecular) gene assembly in ciliates based on the biological foundations set up in [6, 12].

For the future development of this framework, one needs to investigate the theoretical and the experimental aspects of it. Hence, e.g., on the theoretical side various problems concerning the structure and the complexity of (successful) reductions should be investigated. On the experimental side, experiments confirming/rejecting conjectures that certain types of reductions are preferable (e.g., reductions where *ld* operations have “priority”) are very desirable.

**Acknowledgements**

Tero Harju, Ion Petre, and Grzegorz Rozenberg gratefully acknowledge support by TMR Network GETGRATS. David Prescott acknowledges the support of NIGMS Research Grant #GM56161.

**References**