Math. Struct. in Comp. Science (2002), *vol.* 12, *pp.* 113–134. © 2002 Cambridge University Press
DOI: 10.1017/S0960129501003516 Printed in the United Kingdom

String and graph reduction systems for gene assembly in ciliates

ANDRZEJ EHRENFEUCHT[†], ION PETRE[‡]
DAVID M. PRESCOTT[§] and GRZEGORZ ROZENBERG[¶†]

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

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

[§]*University of Colorado, Department of Molecular, Cellular and Developmental Biology, Boulder CO 80309-0347 USA*
Email: prescotd@spot.colorado.edu

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

Received 20 July 2000; revised 12 December 2000

Ciliates have developed a unique nuclear dualism, having two nuclei of different functionality: the germline micronucleus and the somatic macronucleus. The way that ciliates assemble the macronuclear genes after cell mating constitutes one of the most intricate DNA processings in living organisms. This processing is also very interesting from the computational point of view. In this paper, we investigate the operations of loop excision and hairpin excision/reinsertion used in the assembly process. In particular, we consider three levels of formalization of this process, culminating in graph reduction systems.

1. Introduction

DNA computing is a modern, very active, interdisciplinary research area investigating the use of biomolecules for the purpose of computing (see, for example, Adleman (1984), Păun *et al.* (1998) and Condon and Rozenberg (2000)). One line of research investigates DNA computing *in vivo*, that is, within living cells. In particular, the computational nature of the process of assembly of ciliate genes from their micronuclear to their macronuclear form, has attracted the attention of the DNA computing community, see, for example, Landweber and Kari (1998) and Landweber and Kari (1999).

Ciliates have developed a unique nuclear dualism, having two nuclei: the 'storage' micronucleus, which becomes active only in the process of sexual reproduction, and the metabolically active macronucleus, where the expression of genes takes place. After cell mating, ciliates convert a micronuclear genome to a macronuclear genome in a process

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

involving massive and intricate processing of DNA. This process also turns out to be very interesting from the computational point of view.

Some computational aspects of this process were studied in Ehrenfeucht *et al.* (2000), where, in particular, two molecular operations, loop excision and hairpin excision/reinsertion were considered. In this paper we present a *formal* study of the gene assembly process based on these two operations (Ehrenfeucht *et al.* (2000) is an 'informal' paper written for biologists).

In particular, we present three different representations of the structure of a gene (micronuclear, intermediate and macronuclear) with increasing level of abstraction: MDS descriptors, legal words and legal graphs. For each of these representations we present a transformation (rewriting) system modelling the gene assembly process, based on the processing of *pointers*. The most amazing thing about ciliates is that they apparently 'know' the linked list structure, and implement it (during the gene assembly process) in the most elegant way based on pattern matching and molecule folding.

We prove that all three rewriting systems are equivalent in their power of modelling of the gene assembly process. Thus, in particular, the most abstract representation of gene structure, *viz*. through legal graphs, carries enough information to allow the detailed tracing of the gene assembly process.

The paper is organized as follows.

In Section 2 we discuss the basic structure of DNA molecules and introduce our notation for them.

In Section 3 we give a description of the gene assembly process, providing the basic biological intuition for the rest of the paper.

In Section 4 we formalize the structure of a gene through MDS descriptors and formalize in this framework the operations of loop excision and hairpin excision/reinsertion introduced in Section 3. We obtain in this way **ld**-operations and **hi**-operations, respectively, for the rewriting of MDS descriptors.

In Section 5 we introduce legal words, which are obtained by removing certain information from the MDS descriptors. We then introduce the string pointer reduction systems for rewriting legal words, and prove that these systems are equivalent to the MDS descriptor rewriting systems, as far as the strategies for gene assembly are concerned.

In Section 6 legal words are abstracted to legal graphs, and then graph pointer reduction systems for the rewriting of legal graphs are introduced. It is proved then that, although the whole linear structure of legal words is lost when they are abstracted to legal graphs, the graph pointer reduction systems are equivalent to string pointer reduction systems, as far as the strategies for gene assembly are concerned.

In the final section, Section 7 'Discussion', we point out some lines of future research, and compare our model with the model presented in Landweber and Kari (1999).

2. DNA molecules

In this section we give some basic facts about DNA molecules and the notation we will use to denote them. The reader is referred to Prescott (1992a) for more background.

DNA molecules are polymers made out of simple monomers called *nucleotides*. Each nucleotide consists of three components: sugar, phosphate and base. The sugar molecule

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

has five 'attachment' points (carbons) numbered $1'$ through $5'$ with phosphate attached to $5'$ and base attached to $1'$. There are 4 different bases denoted by A, C, G, T, and 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 that binds the $3'$ 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. Note that in such a strand, the nucleotide at one end of the strand has the $5'$ end free (that is, available for bonding), while the nucleotide at the other end of the strand has the $3'$ end 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. As a matter of fact, the $5' - 3'$ direction is 'preferred' by nature, so single-stranded DNA molecules are read in the $5' - 3'$ direction. For example, $ACGTC$ denotes the single-stranded DNA molecule with the nucleotide C at its $3'$ end. Hence, single-stranded DNA molecules are denoted by strings over the alphabet $\{A, C, T, G\}$, where the left-to-right reading orientation of a string corresponds to the $5' - 3'$ polarity of the denoted molecule.

Nucleotides can also bind through their bases by forming a weak (hydrogen) bond. However, such a bond may be formed only between complementary nucleotides: A with T, and C with G (this is referred to as the *Watson–Crick complementarity*). This complementarity is used for forming *double-stranded DNA molecules*, where two single-stranded DNA molecules bind through their bases. However, such a bond may be formed only if the joined single-stranded molecules are of opposite polarity.

We will use the same alphabet of nucleotides $\{A, C, G, T\}$ to denote the double-stranded DNA molecules, by simply writing the strings denoting the two strands of a given DNA molecule above each other. For example, $\begin{matrix} ACGTC \\ TGCAG \end{matrix}$ denotes the double-stranded molecule such that one of its strands is $ACGTC$ and the other is $GACGT$. Thus, in this notation the upper string is in $5' - 3'$ direction, and the lower string is in the $3' - 5'$ direction. This molecule is 'perfect' in the sense that each nucleotide in one strand has a 'matching' nucleotide in the other strand. The molecule (denoted by) $\begin{matrix} ACGTCACT \\ TGCAG \end{matrix}$ is non-perfect, since it has the 'sticky end' ACT with missing complementary nucleotides in the other strand.

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 $\overline{\alpha}$. Hence, the inversion of α is obtained by taking the Watson–Crick complement of the mirror image of α. For example, the inversion of the single-stranded DNA molecule $ACGTC$ is $GACGT$.

In our 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 we denote by the upper string. For example, both $\begin{matrix} ACGTC \\ TGCAG \end{matrix}$ and $\begin{matrix} GACGT \\ CTGCA \end{matrix}$ denote the same molecule: we say that $\begin{matrix} GACGT \\ CTGCA \end{matrix}$ is the

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

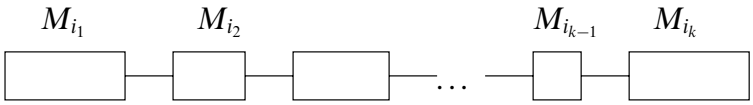


Fig. 1. The structure of a gene in a micronuclear DNA.

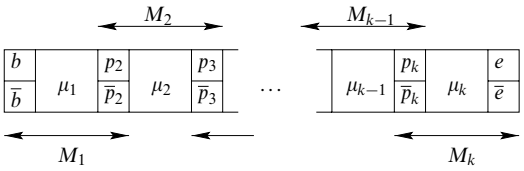


Fig. 2. The structure of a macronuclear assembled gene.

inversion of $\begin{matrix} ACGTC \\ TGCAG \end{matrix}$. Also $\begin{matrix} GACGT \\ TCACTGCA \end{matrix}$ is the inversion of $\begin{matrix} ACGTCACT \\ TGCAG \end{matrix}$. Hence, for a double string α (representing a double-stranded molecule) its inversion is obtained by first exchanging the two single strings of α for each other, and then taking the mirror image of the resulting double string.

3. Gene assembly in ciliates

In ciliates, a gene in a micronuclear chromosome is organized as a sequence of MDSs (macronuclear destined sequences), separated by IESs (internally eliminated sequences) (Prescott 1992b), as sketched in Figure 1, where the rectangles represent MDSs and the connecting line segments between MDSs represent IESs. During the assembly process of a micronuclear gene to its macronuclear form, the IESs are excised and the MDSs are spliced in the orthodox order $M_1 M_2 \ldots M_k$. Each MDS M_i has the structure

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

except for M_1 and M_k, which are of the form

$$M_1 = \left(\begin{matrix} b \\ \overline{b} \end{matrix}, \mu_1, \begin{matrix} p_2 \\ \overline{p}_2 \end{matrix} \right), \quad M_k = \left(\begin{matrix} p_k \\ \overline{p}_k \end{matrix}, \mu_k, \begin{matrix} e \\ \overline{e} \end{matrix} \right),$$

where b designates 'beginning' and e designates 'end'. We refer to each $\begin{matrix} p_i \\ \overline{p}_i \end{matrix}$ as a *pointer*, and to μ_i as the *body* of M_i. Each pointer occurs in an MDS at its boundary: $\begin{matrix} p_i \\ \overline{p}_i \end{matrix}$ is called the *incoming pointer* of M_i, and $\begin{matrix} p_{i+1} \\ \overline{p}_{i+1} \end{matrix}$ is called the *outgoing pointer* of M_i. Thus each pointer has two occurrences: $\begin{matrix} p_i \\ \overline{p}_i \end{matrix}$ is the incoming pointer of M_i and the outgoing pointer of M_{i-1}. Each p_i is in fact a sequence of nucleotides, and $\overline{p}_i$ is the inversion of p_i. On the other hand, b and e are markers – we use them to simply mark the locations where an incipient macronuclear gene will be excised from the micronuclear DNA.

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

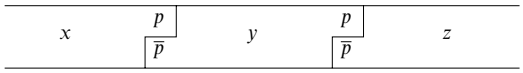


Fig. 3. Direct repeat pattern.

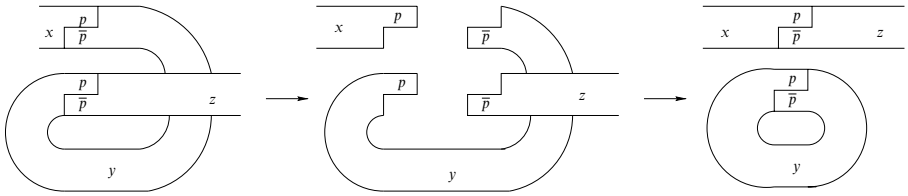


Fig. 4. The *ld*-excision operation.

In the macronuclear gene, the MDSs $M_1, M_2, \ldots, M_k$ are spliced in the way shown in Figure 2: each MDS M_j, $1 \leqslant j < k$, is 'glued' with M_{j+1} on the 'overlapping' pointer $\begin{matrix} p_{j+1} \\ \overline{p}_{j+1} \end{matrix}$.

Two operations contributing to the gene assembly process in ciliates are discussed in Ehrenfeucht *et al.* (2000):

1 the *(loop, direct repeat)-excision* (*ld*-excision, or just *ld*, for short), and
2 the *hairpin, inverted repeat-excision/reinsertion* (*hi*-excision/reinsertion, or just *hi*, for short).

The operation of *ld*-excision is applicable to any molecule having a direct repeat pattern $\left(\begin{matrix} p \\ \overline{p} \end{matrix} , \begin{matrix} p \\ \overline{p} \end{matrix} \right)$ of a pointer $\begin{matrix} p \\ \overline{p} \end{matrix}$; such a molecule is depicted in Figure 3. To apply the operation, the molecule is folded into a loop aligned by the pair of direct repeats, and then the operation proceeds as shown in Figure 4. The excision involves staggered cuts that creates sticky ends, and the operation creates two molecules, a linear one, and a circular one.

An application of the *ld* operation excises a circular molecule. Hence, such an application may only lead to a successful gene assembly 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 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 a *boundary* application). 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, as illustrated in Figure 5.

The operation of hi-excision/reinsertion is applicable to any molecule having an inverted repeat pattern $\left(\begin{matrix} p \\ \overline{p} \end{matrix} , \begin{matrix} \overline{p} \\ p \end{matrix} \right)$ of a pointer $\begin{matrix} p \\ \overline{p} \end{matrix}$; such a molecule is depicted in Figure 6. To apply the operation, the molecule is folded into a hairpin aligned by the inverted repeat pair, and then, the operation proceeds as shown in Figure 7. The excision here involves staggered cuts, but through reinsertion, the operation creates only one molecule.

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

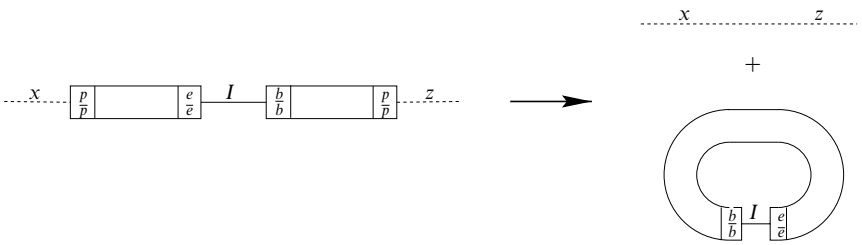


Fig. 5. The boundary application of the *ld*-excision to the pointer 3 results in the assembled gene placed into a circular molecule.

We refer the interested reader to Ehrenfeucht *et al.* (2000; 2001), for a detailed presentation of these gene assembly operations in ciliates.

In this paper we will consider gene assembly from an operational point of view, that is, we will consider possible strategies for gene assembly, where each strategy is a sequence of operations. Each operation is either *ld* or *hi* applied to a specific pair of pointers (either a direct repeat or an inverted repeat, respectively) in a given molecule. This point of view allows us to skip some details concerning the structure of genes. In this way we get three models of DNA processing in ciliates, which will be considered in Sections 4, 5 and 6.

4. MDS structures and MDS descriptors

From the point of view of the gene assembly process, the structural information about the micronuclear or an intermediate precursor τ of a macronuclear gene v can be given by the sequence of MDSs forming τ. The whole process can be thought of as a process of assembling MDSs through splicing to eventually put them in the orthodox order $M_1 \dots M_k$, as shown in Figure 2. Thus, one can represent a macronuclear gene, as well as its micronuclear or an intermediate precursor, by its sequence of MDSs only. We deal in this way with *real* MDS structures. Before formally introducing this notion, we need some notions and notation.

Let τ be a precursor in the gene assembly process of a macronuclear gene v. We will use the alphabet $\mathcal{M} = \{M_{i,j} \mid i, j \in \mathbb{N}^+ \text{ and } i \leqslant j\}$ to denote the MDSs of τ (we use $\mathbb{N}^+$ to denote the set of positive integers). The letters of the form $M_{i,i}$ may also be written as M_i, and they are called *elementary*, while the letters of the form $M_{i,j}$ with $i < j$, are called *composite*. Hence $\mathcal{M} = \mathcal{E} \cup \mathcal{C}$, where $\mathcal{E} = \{M_i \mid i \in \mathbb{N}^+\}$ and $\mathcal{C} = \{M_{i,j} \mid i, j \in \mathbb{N}^+ \text{ and } i < j\}$. The letters from $\mathcal{E}$ are used to denote the MDSs present in the micronuclear precursor of v, referred to as *elementary* MDSs, while the letters from $\mathcal{C}$ are used to denote the *composite* MDSs formed in the process of gene assembly: $M_{i,j}$ denotes the composite MDS that results from splicing the elementary MDSs $M_i, M_{i+1}, \dots, M_j$ by their pointers. Thus, the incoming pointer (or marker) of $M_{i,j}$ is the incoming pointer (or marker) of M_i, and its outgoing pointer (or marker) is the outgoing pointer (or marker) of M_j.

We will use the alphabet $\overline{\mathcal{M}} = \{\overline{M}_{i,j} \mid M_{i,j} \in \mathcal{M}\}$ to denote the inverted MDSs. Let the total alphabet be $\Theta = \mathcal{M} \cup \overline{\mathcal{M}}$. Clearly, Θ is infinite, but for each specific gene v, the set of its elementary MDSs has only a finite number of elements, say k. They will be denoted by

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

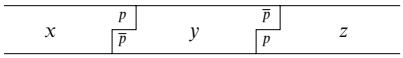


Fig. 6. Inverted repeat pattern.

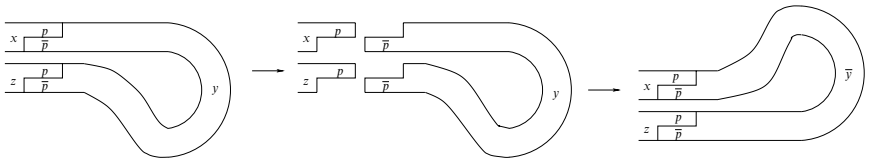


Fig. 7. The hi-excision/reinsertion operation.

the letters from $\{M_1, \ldots, M_k\}$, where the orthodox order of the MDSs in the assembled macronuclear gene is $M_1 M_2 \ldots M_k$ (we say then that v is of MDS *size* k). Thus, we deal with the finite alphabet $\Theta_k = \mathcal{M}_k \cup \overline{\mathcal{M}}_k$, where $\mathcal{M}_k = \{M_{i,j} \mid i, j \in \mathbb{N}^+ \text{ and } i \leqslant j \leqslant k\}$. Unless it is otherwise clear, we assume that k is fixed for the rest of this paper.

To simplify our notation for denoting pointers, we will use in the rest of this paper positive integers $2, 3, \ldots$, rather than $\begin{matrix} p_2 \\ \overline{p}_2 \end{matrix}$, $\begin{matrix} p_3 \\ \overline{p}_3 \end{matrix}$, ..., and $\overline{2}, \overline{3}, \ldots$, rather than $\begin{matrix} \overline{p}_2 \\ p_2 \end{matrix}$, $\begin{matrix} \overline{p}_3 \\ p_3 \end{matrix}$, ... Using this simplified notation, the MDS M_i is $M_i = (i, \mu_i, i+1)$, and its inversion is $\overline{M}_i = \left(\overline{i+1}, \overline{\mu}_i, \overline{i}\right)$. Thus, we need the alphabets $\Delta = \{2, 3, \ldots\}$ and $\overline{\Delta} = \{\overline{2}, \overline{3}, \ldots\}$. We will use $\Pi = \Delta \cup \overline{\Delta}$ to denote the set of *pointers*, and $\Psi = \{b, e, \overline{b}, \overline{e}\}$ to denote the set of *markers*. We will also use the extended sets $\Delta_{ex} = \Delta \cup \{b, e\}$, $\overline{\Delta}_{ex} = \overline{\Delta} \cup \{\overline{b}, \overline{e}\}$, and $\Pi_{ex} = \Pi \cup \Psi$. The 'bar operator' is used to move from Δ_{ex} to $\overline{\Delta}_{ex}$, and the other way around. Hence, for $z \in \Pi_{ex}$, $\overline{z}$ is the *partner* of z from the 'other half of Π_{ex}', and $\overline{\overline{z}} = z$. For any $z \in \Pi$, we call $\{z, \overline{z}\}$ *the pointer set* of z (and of $\overline{z}$), and denote it by **pts**(z) (and **pts**$(\overline{z})$).

We will be quite liberal in our interpretation of formal symbols from Π as pointers in MDSs: sometimes p will mean the double-stranded molecule, and sometimes just one strand of a double-stranded molecule. It is convenient to have such a liberal interpretation, and it should not lead to confusion.

For an MDS $N = (p, \mu, q)$, we say that the pointers (or markers) p and q *occur* in N. For an MDS sequence $x = N_1 \ldots N_l$ and $p \in \Pi_{ex}$, we say that p *occurs* in x if there is $1 \leqslant i \leqslant l$ such that p occurs in N_i.

We generalize now the notion of the orthodox order of elementary MDSs (coming from biology – see, for example, Prescott (1992a)), so that it also includes sequences containing composite MDSs. Thus, a sequence $M_{i_1,j_1} M_{i_2,j_2} \ldots M_{i_n,j_n}$, $n \leqslant k$, is *orthodox* if $i_1 = 1$, $i_l = 1 + j_{l-1}$ for all $2 \leqslant l \leqslant n$ if $n \geqslant 2$, and $j_n = k$. If $i_l = j_l$, for all $1 \leqslant l \leqslant n$, we obtain the orthodox order of elementary MDS sequences.

We then call a sequence over Θ_k a *real MDS structure* if it is a permutation of an orthodox sequence, with possibly some of its elements inverted.

Example 1. The MDS sequence $M_3 \overline{M}_1 \overline{M}_2$ is a real MDS structure, while $M_2 \overline{M}_3 \overline{M}_4$ is not.

In its macronuclear form, a gene no longer has pointers. On the other hand, its

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

micronuclear precursor has pointers that flank IESs. As we observed above, in the process of assembly as long as there are IESs, the MDSs are not yet spliced in the orthodox order, and the assembly process is not yet completed. Thus, as long as there are pointers left, the assembly process is not yet completed. This is the basic idea underlying the analysis of the assembly process by tracing the pointers of the intermediate molecules. Here, the notion of MDS descriptor turns out to be very handy. Intuitively speaking, given a sequence of MDSs corresponding to the micronuclear precursor of a gene or to an intermediate molecule, one denotes each MDS M by the ordered pair of its pointers or markers. If the MDS $M = (p, \mu, q)$, then M is represented as (p, q), and its inversion $\overline{M}$ as $(\overline{q}, \overline{p})$.

More formally, we define the mapping ψ on Θ_k by:

(i) $\psi(M_{1,k}) = (b, e)$, and $\psi(\overline{M}_{1,k}) = (\overline{e}, \overline{b})$,
(ii) $\psi(M_{1,i}) = (b, i+1)$, and $\psi(\overline{M}_{1,i}) = (\overline{i+1}, \overline{b})$, for all $1 \leqslant i < k$,
(iii) $\psi(M_{i,k}) = (i, e)$, and $\psi(\overline{M}_{i,k}) = (\overline{e}, \overline{i})$, for all $1 < i \leqslant k$,
(iv) $\psi(M_{i,j}) = (i, j+1)$, and $\psi(\overline{M}_{i,j}) = (\overline{j+1}, \overline{i})$, for all $1 < i \leqslant j < k$,

where b, e are reserved symbols (b stands for 'beginning', and e for 'end'). Then, for a sequence $X_1 \dots X_l$ over Θ_k,

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

Thus, for example, for the sequence $x = M_{3,5}\overline{M}_{9,11}\overline{M}_{1,2}M_{12}\overline{M}_{6,8}$ and $k = 12$,

$$\psi(x) = (3, 6)(\overline{12}, \overline{9})(\overline{3}, \overline{b})(12, e), (\overline{9}, \overline{6}).$$

Let, for $k > 2$, $\Pi_k = \Delta_k \cup \overline{\Delta}_k$ and $\Pi_{ex,k} = \Delta_k \cup \overline{\Delta}_k \cup \Psi$, where $\Delta_k = \{2, \dots, k-1\}$ and $\overline{\Delta}_k = \{\overline{2}, \dots, \overline{k-1}\}$. Also, let Γ_k be the set consisting of the following ordered pairs:

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

A word over Γ_k is called an MDS *descriptor*.

Clearly, translating a sequence over Θ_k by ψ, we obtain an MDS descriptor. Then, an MDS descriptor is *realistic* if $\delta = \psi(x)$ for some real MDS structure x (over Θ_k).

For any $x = (p, q) \in \Gamma_k$, we say that p is *left* in x and q is *right* in x. For an MDS descriptor δ and $p \in \Pi_{ex,k}$, we say that p has a *left occurrence* (*right occurrence*) in δ, if $\delta = \delta_1 x \delta_2$, for $\delta_1, \delta_2 \in (\Gamma_k)^*$ and $x \in \Gamma_k$, such that p is left (right, respectively) in x. We also say that p *occurs* in δ, if p has either left, or right occurrence in δ.

Recall that for two words w_1, w_2 over an arbitrary alphabet Σ, w_2 is a *subword* of w_1 if $w_1 = uw_2v$, for some words u and v, and w_2 is a *scattered subword* of w_1 if $w_2 = v_1 \dots v_l$, and $w_1 = u_0 v_1 u_1 \dots u_{l-1} v_l u_l$, for some words $v_1, \dots, v_l, u_0, u_1, \dots, u_l$ over Σ.

The following characterization of realistic MDS descriptors will be useful later.

Theorem 1. An MDS descriptor δ is realistic if and only if the following properties hold:

(a) For any marker $m \in \Psi$, δ has exactly one occurrence from the set $\{m, \overline{m}\}$; for any pointer $p \in \Pi_k$, if δ has an occurrence of p, then δ has exactly two occurrences from **pts**(p).

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

(b) For any pointer $p \in \Pi_k$, if p occurs twice in δ, then one occurrence is left and one occurrence is right; if both p and $\overline{p}$ occur in δ, then either both occurrences are left, or both occurrences are right.
(c) No proper scattered subword of δ is a realistic MDS descriptor.

Proof. Assume that δ is a realistic MDS descriptor, $\delta = \psi(x)$, where $x = X_1 \dots X_n$ is a real MDS structure. By the definition of real MDS structures, there is an orthodox sequence $y = M_{i_1,j_1} \dots M_{i_n,j_n}$, and a permutation $(l_1, \dots, l_n)$ of the set $\{1, \dots, n\}$, such that $X_{l_m} \in \{M_{i_m,j_m}, \overline{M}_{i_m,j_m}\}$, for all $1 \leqslant m \leqslant n$.

By the definition of orthodox sequences, $i_1 = 1$, $j_n = k$, and $j_l + 1 = i_{l+1}$, for all $1 \leqslant l < n$. Thus, both markers b and e occur in y, in the MDSs M_{i_1,j_1} and M_{i_n,j_n}, respectively. Moreover, any pointer occurring in y has, in fact, two occurrences: one as the outgoing pointer of some MDS M_{i_l,j_l}, for some $1 \leqslant l < n$, and one as the incoming pointer of the MDS $M_{i_{l+1},j_{l+1}}$. Since x is obtained from y by permuting its elements and possibly inverting some of them, and $\delta = \psi(x)$, we obtain the properties (a) and (b) above.

Assume now that there is a realistic MDS descriptor that is a proper scattered subword of δ. Consequently, there must be an orthodox sequence that is a proper subsequence of $M_{i_1,j_1} M_{i_2,j_2} \dots M_{i_m,j_m}$. This is impossible by the definition of an orthodox sequence.

For the reverse implication of the theorem, let us consider an MDS descriptor δ satisfying the properties (a)–(c) above, $\delta = (p_1, q_1) \dots (p_n, q_n)$. We construct in the following a real MDS structure $x = X_1 \dots X_n$ such that $\delta = \psi(x)$.

By condition (a), either b or $\overline{b}$ occurs in δ, say in (p_{i_1}, q_{i_1}). Since b can only be left and $\overline{b}$ can only be right in the elements of Γ_k, either $p_{i_1} = b$ or $q_{i_1} = \overline{b}$. In the former case we set $r_1 = q_{i_1}$ and $X_{i_1} = M_{1,r_1-1}$, if q_{i_1} is a pointer, and we set $r_1 = e$ and $X_{i_1} = M_{1,k}$, otherwise. In the latter case, we set $r_1 = p_{i_1}$ and $X_{i_1} = \overline{M}_{1,r_1-1}$, if p_{i_1} is a pointer, and we set $r_1 = \overline{e}$ and $X_{i_1} = \overline{M}_{1,k}$, otherwise. Observe that if r_1 is a pointer, then in the former case $r_1 \in \Delta_k$ and r_1 is right in (p_{i_1}, q_{i_1}), and in the latter one, $r_1 \in \overline{\Delta}_k$ and r_1 is left in (p_{i_1}, q_{i_1}).

If r_1 is a pointer, then, by condition (b), δ has two occurrences from $\mathbf{pts}(r_1)$: one in (p_{i_1}, q_{i_1}) as the pointer r_1, and the other say in (p_{i_2}, q_{i_2}), $i_1 \neq i_2$, as the pointer r_1'. We have now two cases: either $r_1' \in \Delta_{ex}$ or $r_1' \in \overline{\Delta}_{ex}$. Consider the former case: $r_1' \in \Delta_{ex}$. Since its partner r_1 is either from Δ_{ex} and right in δ, or from $\overline{\Delta}_{ex}$ and left in δ, r_1' must be left in (p_{i_2}, q_{i_2}): $r_1' = p_{i_2}$. We set $r_2 = q_{i_2}$ and $X_{i_2} = M_{r_1,r_2-1}$ if q_{i_2} is a pointer, and we set $r_2 = e$ and $X_{i_2} = M_{r_1,k}$ otherwise. Similarly, in the latter case, r_1' must be right in (p_{i_2}, q_{i_2}): $r_1' = q_{i_2}$. Then we set $r_2 = p_{i_2}$ and $X_{i_2} = \overline{M}_{r_1,r_2-1}$ if p_{i_2} is a pointer, and we set $r_2 = \overline{e}$ and $X_{i_2} = \overline{M}_{r_1,k}$ otherwise. We continue the same reasoning with r_2.

The procedure stops when we find the ending marker, $r_i \in \mathbf{pts}(e)$, for some $i \geqslant 1$.

Since the set of pointers and markers occurring in δ has $2n$ elements, the procedure will stop after at most n steps. As a matter of fact, the last element to be considered will be the ending marker or its inverse. By condition (c), none of the proper scattered subwords of δ is an MDS descriptor, so the procedures must stop after exactly n steps. Consequently, all the MDSs $X_1, \dots, X_n$ have been defined in our procedure. It is easy to see now that $\psi(x) = \delta$, and thus that δ is a realistic MDS descriptor. □

Corollary 2. For a realistic MDS descriptor δ, if $\delta = \delta_1(x, y)\delta_2$, with x, y markers and $\delta_1, \delta_2 \in (\Gamma_k)^*$, then $\delta_1 = \delta_2 = \Lambda$.

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

Proof. Clearly, (x, y) is a realistic MDS descriptor, which is a scattered subword of δ. By Theorem 1(c), $\delta = (x, y)$, that is, $\delta_1 = \delta_2 = \Lambda$. □

We will now formalize the gene assembling operations *ld* and *hi* through the operations of **ld** and **hi** on realistic MDS descriptors. First, we define the operation of *reversed switch* **rs** on $(\Gamma_k)^*$ as follows:

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

for any $l \geqslant 0$. Then, the operations **ld** and **hi** are defined as follows:

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

$$\mathbf{ld}_p(\delta_1(q, p)(p, r)\delta_2) = \delta_1(q, r)\delta_2,$$
$$\mathbf{ld}_p((p, m_1)(m_2, p)) = (m_2, m_1),$$

   where $q, r \in \Pi_{ex,k}$, $\delta_1, \delta_2 \in (\Gamma_k)^*$, and $m_1, m_2 \in \Psi$. The first type of application of $\mathbf{ld}_p$ is called *simple* and the second one *boundary*.
2. For each $p \in \Pi_k$, the **hi**-operation for p, denoted by $\mathbf{hi}_p$, is defined as follows:

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

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

Let $\mathbf{LD}_{\Pi_k} = \{\mathbf{ld}_p \mid p \in \Pi_k\}$ be the set of **ld**-operations *over* Π_k, and $\mathbf{HI}_{\Pi_k} = \{\mathbf{hi}_p \mid p \in \Pi_k\}$ be the set of **hi**-operations *over* Π_k, and $\mathbf{RL}_{\Pi_k} = \mathbf{LD}_{\Pi_k} \cup \mathbf{HI}_{\Pi_k}$ be the set of *operations* over Π_k. We may skip the subscript Π_k and write **LD**, **HI**, and **RL** whenever Π_k is clear from the context.

For a realistic MDS descriptor δ and operations $\rho_1, \dots, \rho_l$, $l \geqslant 1$, from $\mathbf{RL}_{\Pi_k}$,

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

is a *strategy* for δ (*based on* $\{\mathbf{ld}, \mathbf{hi}\}$) if ρ_1 is applicable to δ, and ρ_i is applicable to $\rho_{i-1}\dots\rho_1(\delta)$, for all $1 < i \leqslant l$. Such a strategy is *successful* if $\rho_l\dots\rho_1(\delta)$ is either (b, e), or $(\overline{e}, \overline{b})$.

Example 2. Consider the realistic MDS descriptor

$$\delta = (5, e)(b, 2)(2, 3)(\overline{5}, \overline{4})(3, 4).$$

The operation $\mathbf{ld}_2$ is applicable to δ, and $\mathbf{ld}_2(\delta) = (5, e)(b, 3)(\overline{5}, \overline{4})(3, 4)$. Also, the operation $\mathbf{hi}_5$ is applicable to δ, and $\mathbf{hi}_5(\delta) = (\overline{3}, \overline{2})(\overline{2}, \overline{b})(\overline{e}, \overline{4})(3, 4)$.

The strategy $(\delta; \mathbf{ld}_2, \mathbf{hi}_4, \mathbf{hi}_3, \mathbf{ld}_5)$ is a successful strategy for δ, and it yields the following sequence of realistic MDS descriptors:

$$\mathbf{ld}_2(\delta) = (5, e)(b, 3)(\overline{5}, \overline{4})(3, 4),$$
$$\mathbf{hi}_4\,\mathbf{ld}_2(\delta) = (5, e)(b, 3)(\overline{5}, \overline{3}),$$
$$\mathbf{hi}_3\,\mathbf{hi}_4\,\mathbf{ld}_2(\delta) = (5, e)(b, 5),$$
$$\mathbf{ld}_5\,\mathbf{hi}_3\,\mathbf{hi}_4\,\mathbf{ld}_2(\delta) = (b, e).$$

In this section we have formalized the operational strategies for gene assembly in the framework of realistic MDS descriptors. The main simplification leading to this framework

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

was to dispose of IESs and of bodies of MDSs – they are irrelevant from our operational point of view.

5. String pointer reduction systems

In an MDS descriptor, the information about the modelled molecule consists of the sequence of the incoming and the outgoing pointers and markers. However, one can trace the intermediate molecules, as well as the assembly process, just by tracing the sequence of pointers in the molecule. Consequently, we are able to simplify our formal system by removing the parenthesis from the realistic MDS descriptors and deleting the markers. We deal in this way with legal words.

A *legal word over* Π_k (or simply a *legal word* if Π_k is understood from the context) is a word $\pi \in \Pi_k^*$ such that for each $p \in \Pi_k$, if $\mathbf{alph}(\pi) \cap \mathbf{pts}(p) \neq \varnothing$, then π contains exactly two occurrences from $\mathbf{pts}(p)$. Thus, either π contains one occurrence of p and one occurrence of $\overline{p}$, in which case $\mathbf{pts}(p)$ is said to be *good* in π, or it contains two occurrences of p, or two occurrences of $\overline{p}$, in which cases $\mathbf{pts}(p)$ is said to be *bad* in π.

For a realistic MDS descriptor $\delta = (p_1, q_1) \dots (p_m, q_m)$, by writing just the sequence of pointers and deleting the markers, we obtain the *legal word* π_δ *corresponding to* δ.

For a word $\pi = x_1 x_2 \dots x_n \in \Pi_k^*$, the *reversed switch* of π, denoted $\mathbf{rs}(\pi)$, is the word $\overline{x}_n \dots \overline{x}_2 \overline{x}_1$ (recall that for each $x \in \Pi_k$, $\overline{\overline{x}} = x$). For a pointer $p \in \Pi_k$ such that $\{x_i, x_j\} \subseteq \mathbf{pts}(p)$, for some positive integers $i < j \leqslant n$, the *p-interval* of π is the subword $x_i x_{i+1} \dots x_j$. We say that two pointers $p, q \in \Pi_k$ *overlap in* π if the p-interval of π overlaps with the q-interval of π – we also say then that $\mathbf{pts}(p)$ overlaps with $\mathbf{pts}(q)$ (recall that for $i_2 \geqslant i_1$, the subwords $x_{i_1} \dots x_{j_1}$ and $x_{i_2} \dots x_{j_2}$ of π overlap iff $i_2 \leqslant j_1$ and $j_1 < j_2$).

Example 3. The legal word π_δ corresponding to the realistic MDS descriptor $\delta = (5, e)(b, 2)(2, 3)(\overline{5}, \overline{4})(3, 4)$ is $\pi_\delta = 5\,2\,2\,3\,\overline{5}\,\overline{4}\,3\,4$. The 2-interval of π_δ is the subword 22, the 3-interval of π_δ is the subword $3\overline{5}\overline{4}3$, the 4-interval is $\overline{4}34$, and the 5-interval is $5223\overline{5}$. The pointer 3 overlaps in π_δ with both 4 and 5, but not with 2.

We will introduce now a *string pointer reduction system* over Π_k (SPRS_{Π_k} for short, or simply SPRS, when Π_k is understood from the context), which is a formal system modelling the transformation of pointers when **ld** and **hi** operations are used in the gene assembly process. As a matter of fact, SPRS defines reductions of legal words over Π_k in such a way that each reduction removes all the occurrences of one pointer set. SPRS_{Π_k} consists of two sets of reduction rules:

1 For each $p \in \Pi_k$, the string negative rule for p, denoted by $\mathbf{snr}_p$, is the rule of the form

$$pp \to \Lambda.$$

2 For each $p \in \Pi_k$, the string positive rule for p, denoted by $\mathbf{spr}_p$, is the rule of the form

$$p\overline{p} \to \Lambda.$$

Let $\mathrm{SNR}_{\Pi_k} = \{\mathbf{snr}_p \mid p \in \Pi_k\}$ be the set of string negative rules *over* Π_k, and $\mathrm{SPR}_{\Pi_k} = \{\mathbf{spr}_p \mid p \in \Pi_k\}$ be the set of string positive rules *over* Π_k; thus the set of all

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

string rules $\mathrm{SPRS}_{\Pi_k} = \mathrm{SNR}_{\Pi_k} \cup \mathrm{SPR}_{\Pi_k}$. We may omit the subscript Π_k and write SNR and SPR whenever Π_k is clear from the context.

Let π be a legal word over Π_k and let $p \in \Pi_k$. The rule $\mathbf{snr}_p$ is *applicable* to π if pp is a *subword* of π, that is, π is of the form $\pi = \pi_1 pp \pi_2$, for some π_1, π_2 words over Π_k. The result of the application of $\mathbf{snr}_p$ to π is the legal word $\mathbf{snr}_p(\pi) = \pi_1\pi_2$.

The rule $\mathbf{spr}_p$ is *applicable* to π if $p\overline{p}$ is a *scattered subword* of π (that is, if $\pi = \pi_1 p \pi_2 \overline{p} \pi_3$, for some words π_1, π_2, π_3 over Π_k). The result of the *direct reduction* of π by $\mathbf{spr}_p$ is the legal word $\mathbf{spr}_p(\pi) = \pi_1\, \mathbf{rs}(\pi_2)\pi_3$.

Since, for each pointer $p \in \Pi_k$, each legal word π having an occurrence of p has exactly two occurrences from $\mathbf{pts}(p)$, there is at most one application of either $\mathbf{snr}_p$ or $\mathbf{spr}_p$ to π.

For a legal word π and a rule ρ from SPRS, $D = (\pi; \rho)$ is a *direct reduction of* π by ρ if ρ is applicable to π. The *result of* D is the legal word $\rho(\pi)$.

For a legal word π and a sequence of reduction rules $\rho_1, \ldots, \rho_l$, $l \geqslant 1$, from SPRS, $D = (\pi; \rho_1, \ldots, \rho_l)$ is a *reduction of* π by $\rho_1, \ldots, \rho_l$ if ρ_1 is applicable to π, and ρ_i is applicable to $\rho_{i-1} \ldots \rho_1(\pi)$, for all $1 < i \leqslant l$. The *result of* D is the legal word $\rho_l \ldots \rho_1(\pi)$. We say that D is *successful* if the result of D is the empty word – we also say then that π is *successful* (in SPRS).

Note that it follows directly from the above definition of reduction that $\rho_i \neq \rho_j$ whenever $i \neq j$.

Example 4. The word $\delta = 5\,2\,2\,3\,\overline{5}\,\overline{4}\,3\,4$ is legal. The rule $\mathbf{snr}_2$ is applicable to δ, and $\mathbf{snr}_2(\delta) = 5\,3\,\overline{5}\,\overline{4}\,3\,4$. Also, the rule $\mathbf{spr}_5$ is applicable to δ, and $\mathbf{spr}_5(\delta) = \overline{3}\,\overline{2}\,\overline{2}\,\overline{4}\,3\,4$.

The reduction $(\delta; \mathbf{snr}_2, \mathbf{spr}_4, \mathbf{spr}_3, \mathbf{snr}_5)$ is a successful reduction, so δ is successful (in SPRS).

Note that we use the 'bar' notation for denoting the inversion of a (single-stranded or a double-stranded) DNA molecule, for example, for a double stranded body μ_i of M_i, we use $\overline{\mu}_i$ to denote its inversion. We also use the 'bar' notation for formal symbols denoting pointers in our pointer reduction systems, but this should not lead to confusion: after all, in the 'intended semantics', if p denotes a pointer (a string of nucleotides), $\overline{p}$ denotes its inversion.

The effect of *ld* and *hi* operations on DNA molecules was translated in the previous section into a formal system, transforming realistic MDS descriptors through the use of **ld** and **hi** operations. We will now relate this formal system to SPRS, which operates on legal words. In particular, we prove that a realistic MDS descriptor δ has a successful strategy (based on $\{\mathbf{ld}, \mathbf{hi}\}$) if and only if the legal word π_δ is successful (in SPRS).

We consider first the **ld** operation.

Lemma 3. Let δ be a realistic MDS descriptor and let $p \in \Pi_k$. If $\mathbf{ld}_p$ is applicable to δ, then $\mathbf{snr}_p$ is applicable to π_δ. Moreover, if $\delta' = \mathbf{ld}_p(\delta)$, and $\pi' = \mathbf{snr}_p(\pi_\delta)$, then $\pi' = \pi_{\delta'}$.

Proof. If $\mathbf{ld}_p$ is applicable to δ, this application is either simple or boundary. In both cases pp is a subword of π_δ, so $\mathbf{snr}_p$ is applicable to π_δ.

Applying $\mathbf{ld}_p$ to δ removes p (and a bigger composite MDS is formed), but in the resulting $\delta' = \mathbf{ld}_p(\delta)$ the sequence of the remaining pointers is the same as in δ.

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

Applying $\mathbf{snr}_p$ to π_δ removes both occurrences of p in π_δ, yielding π'. But the sequence of the remaining pointers in π' is the same as in π_δ.

Consequently, $\pi' = \pi_{\delta'}$ □

Lemma 4. Let δ be a realistic MDS descriptor and let $p \in \Pi_k$. If $\mathbf{snr}_p$ is applicable to π_δ, then $\mathbf{ld}_p$ is applicable to δ. Moreover, if $\delta' = \mathbf{ld}_p(\delta)$, and $\pi' = \mathbf{snr}_p(\pi_\delta)$, then $\pi' = \pi_{\delta'}$.

Proof. If $\mathbf{snr}_p$ is applicable to π_δ, then pp is a subword of π_δ. By construction, it follows immediately that only markers may be present between the two occurrences of p in δ. We consider the following three cases.

Case 1. There are no markers in δ between the two occurrences of p. Since δ is a realistic MDS descriptor, we have $\delta = \delta_1(q_1, p)(p, q_2)\delta_2$, for some $q_1, q_2 \in \Pi_{ex,k}$, and $\delta_1, \delta_2 \in (\Gamma_k)^*$. Clearly, $\mathbf{ld}_p$ is applicable to δ.

Case 2. There is one marker in δ between the two occurrences of p. Hence, we have either $\delta = \delta_1(p, q)(p, r)\delta_2$ or $\delta = \delta_1(r, p)(q, p)\delta_2$ for some $r \in \Pi_{ex,k}$, $q \in \Psi$ and $\delta_1, \delta_2 \in (\Gamma_k)^*$. However, in both cases either both occurrences of p are left, or both occurrences of p are right. By Theorem 1, this case cannot happen.

Case 3. There are two markers in δ between the two occurrences of p. We have three subcases here.

Case 3.1. $\delta = (p, r)(s, p)$ for some $r, s \in \Psi$. Clearly, $\mathbf{ld}_p$ is applicable to δ, yielding (s, r).

Case 3.2. $\delta = \delta_1(p, r)(s, p)\delta_2$ for some $r, s \in \Psi$, $\delta_1, \delta_2 \in (\Gamma_k)^*$, such that $\delta_1\delta_2 \neq \Lambda$. In this case, $(p, r)(s, p)$ is a realistic MDS descriptor that is a proper scattered subword of δ. Since δ is a realistic MDS descriptor, Theorem 1(c) implies that this case is not possible.

Case 3.3. $\delta = \delta_1(q_1, p)(r, s)(p, q_2)\delta_2$ for some $\delta_1, \delta_2 \in (\Gamma_k)^*$, $r, s \in \Psi$, and $q_1, q_2 \in \Pi_k$. This contradicts Corollary 2, so this case is also not possible.

Hence, only Case 1 and Case 3.1 are possible, and in both cases $\mathbf{ld}_p$ is applicable. Now, by Lemma 3, we have $\pi' = \pi_{\delta'}$. □

From Lemmas 3 and 4, we obtain the intertranslatability between the operation **ld** on realistic MDS descriptors and the reduction rule **snr** on legal words.

Corollary 5. Let δ be a realistic MDS descriptor, and let $p \in \Pi_k$. The rule $\mathbf{ld}_p$ is applicable to δ if and only if $\mathbf{snr}_p$ is applicable to π_δ. Moreover, in the case of applicability, if $\pi' = \mathbf{snr}_p(\pi_\delta)$ and $\delta' = \mathbf{ld}_p(\delta)$, then $\pi' = \pi_{\delta'}$.

We now consider the **hi** operation.

Lemma 6. Let δ be a realistic MDS descriptor, and let $p \in \Pi_k$. If $\mathbf{hi}_p$ is applicable to δ, we have $\mathbf{spr}_p$ is applicable to π_δ. Moreover, if $\delta' = \mathbf{hi}_p(\delta)$ and $\pi' = \mathbf{spr}_p(\pi_\delta)$, then $\pi' = \pi_{\delta'}$.

Proof. If $\mathbf{hi}_p$ is applicable to δ, then both p and $\overline{p}$ occur in δ. By Theorem 1, they are either both left, or both right in δ. We then have two situations: either the pointer p occurs in δ before $\overline{p}$, or the other way around. We will only consider the former case, since the reasoning for the latter case is analogous.

Since $\mathbf{hi}_p$ is applicable to δ, $p\overline{p}$ is a scattered subword of π_δ, so the reduction rule $\mathbf{spr}_p$ is applicable to π_δ.

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

If $\delta = \delta_1(p, q_1)(q_2, q_3)\dots(q_{2i}, q_{2i+1})(\overline{p}, r)\delta_2$, for some $q_1, \dots, q_{2i+1}, r \in \Pi_{ex,k}$, and $\delta_1, \delta_2 \in (\Gamma_k)^*$, then $\delta' = \delta_1(\overline{q}_{2i+1}\overline{q}_{2i})\dots(\overline{q}_3, \overline{q}_2)(\overline{q}_1, r)\delta_2$.

Similarly, if $\delta = \delta_1(r, p)(q_1, q_2)\dots(q_{2i-1}, q_{2i})(q_{2i+1}, \overline{p})\delta_2$, for some $\delta_1, \delta_2 \in (\Gamma_k)^*$ and $q_1, \dots, q_{2i+1}, r \in \Pi_{ex,k}$, then

$$\delta' = \delta_1(r, \overline{q}_{2i+1})(\overline{q}_{2i}, \overline{q}_{2i-1})\dots(\overline{q}_2, \overline{q}_1)\delta_2.$$

Thus, to obtain δ' from δ using $\mathbf{hi}_p$, p and $\overline{p}$ are removed, all the pointers in the p-interval of δ are switched, and their sequence is reversed, while the pointers outside the p-interval remain unchanged.

On the other hand, to obtain π' from π_δ using $\mathbf{spr}_p$, p and $\overline{p}$ are removed, all pointers in the p-interval are switched, and their sequence reversed.

Consequently, $\pi_{\delta'} = \pi'$. □

Lemma 7. Let δ be a realistic MDS descriptor, and let $p \in \Pi_k$. If $\mathbf{spr}_p$ is applicable to π_δ, then $\mathbf{hi}_p$ is applicable to δ. Moreover, if $\delta' = \mathbf{hi}_p(\delta)$ and $\pi' = \mathbf{spr}_p(\pi_\delta)$, then $\pi' = \pi_{\delta'}$.

Proof. Since $\mathbf{spr}_p$ is applicable to π_δ, $p\overline{p}$ is a scattered subword of π_δ; in particular, both p and $\overline{p}$ occur in δ. Moreover, by Theorem 1, either both p and $\overline{p}$ are left pointers in δ, or they are both right pointers in δ. In both cases, the rule $\mathbf{hi}_p$ is applicable to δ.

Now, by Lemma 6, we have $\pi' = \pi_{\delta'}$. □

Lemmas 6 and 7 yield the intertranslatability between the operation **hi** on realistic MDS descriptors and the reduction rule **spr** on legal words.

Corollary 8. Let δ be a realistic MDS descriptor, and let $p \in \Pi_k$. The rule $\mathbf{hi}_p$ is applicable to δ if and only if $\mathbf{spr}_p$ is applicable to π_δ. Moreover, in the case of applicability, if $\pi' = \mathbf{spr}_p(\pi_\delta)$ and $\delta' = \mathbf{hi}_p(\delta)$, then $\pi' = \pi_{\delta'}$.

The main result of this section now follows from Corollaries 5 and 8.

Theorem 9. A realistic MDS descriptor δ has a successful strategy if and only if the legal string π_δ is successful in SPRS.

6. Graph pointer reduction systems

We move now to consider the graph pointer reduction system (GPRS for short) and its use in modelling the gene assembly process based on **ld** and **hi** operations. As a matter of fact, we relate GPRS to SPRS, and then, using the results of the previous section, we prove that a realistic MDS descriptor has a successful strategy based on $\{\mathbf{ld}, \mathbf{hi}\}$ if and only if its associated graph is reducible to the empty graph in GPRS.

A *graph* is an ordered pair $G = (V, E)$, where V is a finite set of *vertices*, and $E \subseteq \{\{u, v\} \mid u, v \in V, u \neq v\}$ is a set of *edges*. The graph $(\varnothing, \varnothing)$ is called the *empty graph* and is denoted by $\varnothing$. The *complement* of G, denoted by $\overline{G}$, is the graph $(V, \overline{E})$, where $\overline{E} = \{\{u, v\} \mid u, v \in V, u \neq v, \{u, v\} \notin E\}$.

The vertices $u, v \in V$ are *neighbours* in G if $\{u, v\} \in E$. For a vertex $u \in V$, the *neighbourhood* of u in G is the set of all neighbours of v, that is, the set $\{v \in V \mid \{u, v\} \in E\}$. The vertex v is called *isolated* if its neighbourhood is empty.

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

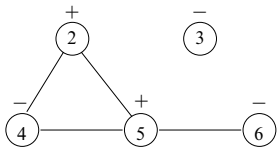


Fig. 8. The graph γ_π associated to $\pi = 2\,6\,5\,6\,\overline{4}\,\overline{3}\,\overline{3}\,\overline{2}\,\overline{5}\,\overline{4}$.

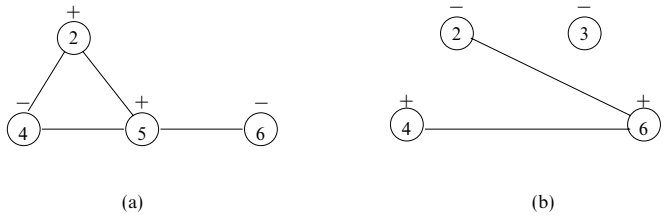


Fig. 9. The reduction of the graph in Figure 8 by: (a) $\mathbf{gnr}_3$, and (b) $\mathbf{gpr}_5$.

For a set of vertices $U \subseteq V$, the *subgraph induced* by U in G is the graph (U, E_U), where $E_U = \{\{u,v\} \in E \mid u,v \in U\}$.

A *labelled graph* is a structure $G = (V, E, \Theta, \theta)$, where (V, E) is a graph, Θ is a set of *labels*, and $\theta : V \to \Theta$ is the *labelling function* of G.

A *signed graph* is a labelled graph $G = (V, E, \Theta, \theta)$, where $\Theta = \{+, -\}$. The vertices labelled by '+' are called *positive*, and the vertices labelled by '−' are called *negative*.

The complement of a signed graph $G = (V, E, \{+,-\}, \theta)$ is the signed graph $\overline{G} = (V, \overline{E}, \{+,-\}, \overline{\theta})$, where $(V, \overline{E})$ is the complement of the graph (V, E), and, for all $v \in V$, $\overline{\theta}(v) = +$ if and only if $\theta(v) = -$. Here we again use the bar notation for denoting the complement of a graph. There is no way that this can be confused with barring pointers or other notations for DNA molecules to denote the inversion. Since the bar notation for denoting the complement of a graph is so standard, we have decided to use it also in this paper.

Let π be a legal word, and let $\mathcal{P}(\pi) = \{\{p, \overline{p}\} \mid \{p, \overline{p}\} \cap \mathbf{alph}(\pi) \neq \varnothing\}$; hence, $\mathcal{P}(\pi)$ consists of the pointer sets of all pointers occurring in π. Then we associate with π the signed graph

$$\gamma_\pi = (V_\pi, E_\pi, \{+,-\}, \theta_\pi),$$

such that $V_\pi = \mathcal{P}(\pi)$, $E_\pi = \{\{P, Q\} \mid P, Q \text{ overlap in } \pi\}$, and, for each $P \in V_\pi$, we have $\theta_\pi(P) = +$ if and only if P is good in π.

We say that γ_π is the *legal graph* of π. Hence, in the terminology of Harary (1969), our legal graphs are *overlap graphs* and, also, *signed circle graphs.*

Example 5. The graph γ_π for the legal word $\pi = 2\,6\,5\,6\,\overline{4}\,\overline{3}\,\overline{3}\,\overline{2}\,\overline{5}\,\overline{4}$ is shown in Figure 8. In pictorial representation of legal graphs, a vertex $\mathbf{pts}(p)$ is represented by either p, or $\overline{p}$.

The *graph pointer reduction system* GPRS *over* Π_k (GPRS_{Π_k} for short, or simply GPRS, when Π_k is understood from the context), defines reductions of legal graphs in such a way

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

that each reduction removes one vertex of the graph (hence one pointer set). GPRS_{Π_k} consists of two sets of *reduction rules.*

1 For any pointer set P, the graph negative rule for P, denoted by $\mathbf{gnr}_P$, is the rule of the form

$$(P, -) \to \varnothing.$$

2 For any pointer set P, the graph positive rule for P, denoted by $\mathbf{gpr}_P$, is the rule of the form

$$(P, +) \to \varnothing.$$

Let $\mathrm{GNR}_{\Pi_k} = \{\mathbf{gnr}_P \mid P \text{ is a pointer set over } \Pi_k\}$ be the set of graph negative rules *over* Π_k, and let $\mathrm{GPR}_{\Pi_k} = \{\mathbf{gpr}_P \mid P \text{ is a pointer set over } \Pi_k\}$ be the set of graph positive rules *over* Π_k; thus $\mathrm{GPRS}_{\Pi_k} = \mathrm{GNR}_{\Pi_k} \cup \mathrm{GPR}_{\Pi_k}$. We may omit the subscript Π_k and write GNR and GPR, whenever Π_k is clear from the context.

Let γ be a legal graph and P a vertex of γ.

The rule $\mathbf{gnr}_P$ is *applicable* to γ if P is a negative isolated vertex of γ. The result of the application of $\mathbf{gpr}_P$ to γ, denoted $\mathbf{gpr}_P(\gamma)$, is the graph obtained from γ by removing (*reducing*) the vertex P.

The rule $\mathbf{gpr}_P$ is *applicable* to γ if P is a positive vertex of γ. The result of the application of $\mathbf{gpr}_P$ to γ, denoted $\mathbf{gpr}_P(\gamma)$, is the graph obtained from γ by removing (reducing) the vertex P (together with all its adjacent edges), and changing the subgraph induced by the neighbourhood of P in γ into its complement.

For a legal graph γ and a rule ρ from GPRS, $D = (\gamma; \rho)$ is a *direct reduction* (of γ *by* ρ) if ρ is applicable to γ. The *result of* D is the legal graph $\rho(\gamma)$.

For a legal graph γ, and a sequence $\rho_1, \ldots, \rho_l$, $l \geqslant 1$, of reduction rules from GPRS, $D = (\gamma; \rho_1, \ldots, \rho_l)$ is a *reduction* of γ if ρ_1 is applicable to γ, and ρ_i is applicable to $\rho_{i-1} \ldots \rho_1(\gamma)$, for all $2 \leqslant i \leqslant l$. The *result* of D is the legal graph $\rho_l \ldots \rho_1(\gamma)$. We say that D is *successful* if the result of D is the empty graph – we also say then that γ is *successful* (in GPRS).

Note that it follows directly from the definition of reduction that $\rho_i \neq \rho_j$ for $i \neq j$.

Example 6. The rules $\mathbf{gnr}_3$ and $\mathbf{gpr}_5$ are both applicable to the graph γ_π from Figure 8. The results of direct reductions of γ_π using the two rules are shown in Figure 9. The graph γ_π is successful, and $(\gamma_\pi; \mathbf{gnr}_3, \mathbf{gpr}_5, \mathbf{gpr}_6, \mathbf{gpr}_2, \mathbf{gpr}_4)$ is a successful reduction.

Using the terminology introduced above, we investigate now the relationship between the string pointer reduction system SPRS and the graph pointer reduction system GPRS. More precisely, we will prove that a legal word π is successful (in SPRS) if and only if γ_π is successful (in GPRS). Moreover, we will show that any reduction of π in SPRS can be translated into a reduction of γ_π in GPRS, and the other way around.

In order to simplify the notation, in the following we will use lower case letters p, q, r to denote pointers (from π) and the upper case letters P, Q, and R to denote the pointer sets of p, q, and r, respectively.

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

Lemma 10. Let π be a legal word, and let $p \in \Pi_k$. If $\mathbf{snr}_p$ is applicable to π, then $\mathbf{gnr}_P$ is applicable to γ_π. Moreover, if $\pi' = \mathbf{snr}_p(\pi)$, and $\gamma' = \mathbf{gnr}_P(\gamma_\pi)$, then $\gamma_{\pi'} = \gamma'$.

Proof. If $\mathbf{snr}_p$ is applicable to π, then p has two occurrences in π, so p is negative in π. Thus, P is negative in γ_π. Moreover, since p does not overlap with any other pointer in π, P is isolated in γ_π. Consequently, $\mathbf{gnr}_P$ is applicable to γ_π.

To prove the second claim of the lemma, we note that $\mathbf{snr}_\pi$ removes the two (consecutive) occurrences of p in π, and the overlap situation for all other pointers occurring in π remains the same as in π'. On the other hand, $\mathbf{gnr}_P$ removes only the isolated vertex P from γ_π, and the resulting graph γ' has exactly the same edges (between remaining vertices) and the same labels (of the remaining vertices).

Consequently, $\gamma_{\pi'} = \gamma'$. □

Lemma 11. Let π be a legal word, and let $p \in \Pi_k$. If $\mathbf{spr}_p$ is applicable to π, then $\mathbf{gpr}_P$ is applicable to γ_π. Moreover, if $\pi' = \mathbf{spr}_p(\pi)$, and $\gamma' = \mathbf{gpr}_P(\gamma_\pi)$, then $\gamma_{\pi'} = \gamma'$.

Proof. If $\mathbf{spr}_p$ is applicable to π, then p is a positive pointer in π, so the vertex P is positive in γ_π. Consequently, $\mathbf{gpr}_P$ is applicable to γ_π.

To prove the second claim of the lemma, we now consider two arbitrary pointers q and r from $\mathbf{alph}(\pi)$, such that p, q, r are three different pointers. We will consider separately each of the three possible overlap situations (of q and r with p):

1 Neither q nor r overlap with p.
2 Only one of q, r overlaps with p.
3 Both q and r overlap with p.

ad 1. Since neither q, nor r overlap with p in π, the application of $\mathbf{spr}_p$ does not change the overlap relationship between q and r: it is the same in π as it is in π'. Moreover, the goodness of q and r remains the same. Since Q and R are not neighbours of P in γ_π, the edge relationship between Q and R in γ' is the same as in γ_π, and their labels remain the same. Thus, the second claim of the lemma follows.

ad 2. Assume that q overlaps with p in π while r does not (the other case is symmetric). This implies that Q is a neighbour of P in γ_π, but R is not. Consequently, the edge relationship between Q and R is the same in γ' as in γ_π. However, the labelling of Q is switched, while the labelling of R is the same in γ' as in γ_π.

Since q overlaps with p in π, one of the two q-partners (say q_1) occurs within the p-interval and the other (say q_2), occurs outside the p-interval. Because of the overlap, applying $\mathbf{spr}_p$ to π changes the goodness of $\mathbf{pts}(q)$.

(i) Assume now that q and r do not overlap in π. Hence, either both q_1 and q_2 are outside the r-interval, or they are both inside it. Since the r-interval must be either within, or outside the p-interval (p and r do not overlap), also in π', after the application of $\mathbf{spr}_p$, q and r do not overlap.
(ii) If q and r overlap in π, then either q_2 must occur within the r-interval, which must then lie outside the p-interval, or q_1 must occur within the r-interval, which must then lie inside the p-interval. In both cases, also in π', after the application of $\mathbf{spr}_p$, q and r

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

overlap. Hence, the application of $\mathbf{spr}_p$ to π does not change the overlap relationship between q and r: it is the same in π as in π'.

Thus, in both cases (*i*) and (*ii*), we have $\gamma' = \gamma_{\pi'}$.

ad 3. In this case, Q and R are both in the neighbourhood of P and thus, after applying $\mathbf{gpr}_P$ to γ_π, their labelling, as well as their edge relationship is changed.

Since both q and r overlap with p, both $\mathbf{pts}(q)$ and $\mathbf{pts}(r)$ has one occurrence inside the p-interval and one occurrence outside the p-interval. Thus, the application of $\mathbf{spr}_p$ to π changes the goodness of both q and r.

For the outside occurrences of $\mathbf{pts}(q)$ and $\mathbf{pts}(r)$, say q_1 and r_1, we have two possible cases (let $\pi = \pi_1\pi_2\pi_3$, where π_2 is the p-interval):

(a) q_1 and r_1 are on the opposite sides of π_2, that is, one of q_1, r_1 is in π_1 and the other one in π_3,

(b) q_1 and r_1 are on the same side of the π_2, that is, either both q_1 and r_1 are in π_1, or both q_1 and r_1 are in π_3.

In both cases, applying $\mathbf{spr}_p$ to π changes the overlap relationship of q and r. Hence, the edge relationship between q and r in $\gamma_{\pi'}$ is changed with respect to γ_π, and thus, it is the same as in γ'.

Consequently, the graphs γ' and $\gamma_{\pi'}$ have the same edges, and the same vertex labels, and so $\gamma' = \gamma_{\pi'}$. Hence, the lemma holds.

Thus, in all three cases, $\gamma_{\pi'} = \gamma'$, and the lemma holds. □

Lemmas 10 and 11 together yield the following corollary.

Corollary 12. Every string reduction $(\pi; \rho_1, \ldots, \rho_l)$ in SPRS translates into a graph reduction $(\gamma_\pi; \rho'_1, \ldots, \rho'_l)$ in GPRS, by translating:

— each SPRS reduction rule $\mathbf{snr}_p$ into the GPRS reduction rule $\mathbf{gnr}_P$;
— each SPRS reduction rule $\mathbf{spr}_p$ into the GPRS reduction rule $\mathbf{gpr}_P$.

Consequently, if π is a successful legal word, then γ_π is a successful legal 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 say that a reduction (either in SPRS, or in GPRS) is *canonical* if all the positive pointers (vertices) are reduced before the reduction of any negative pointer (vertex).

The following two results show that any successful reduction in either SPRS, or GPRS can be modified into a canonical successful reduction.

Lemma 13. If π is a successful legal word in SPRS, then π has a canonical successful reduction in SPRS.

Proof. Let π be a successful legal word, and let $(\pi, \rho_1, \ldots, \rho_l)$ be a successful reduction of π in SPRS. Assume that for some $1 \leqslant i \leqslant l$, $\rho_i = \mathbf{snr}_q$ and $\rho_{i+1} = \mathbf{spr}_p$, for some pointers p, q. If the (consecutive) occurrences of q in π are outside the p-interval of π, replacing $\mathbf{snr}_q$ by $\mathbf{spr}_p$ and $\mathbf{spr}_p$ by $\mathbf{snr}_q$ still gives a successful reduction of π. On the other

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

hand, if the occurrences of q in π are inside the p-interval of π, replacing $\mathbf{snr}_q$ by $\mathbf{spr}_p$ and $\mathbf{spr}_p$ by $\mathbf{snr}_{\overline{q}}$ yields a successful reduction of π in SPRS. □

Lemma 14. If γ is a successful legal graph in GPRS, then γ has a canonical successful reduction in GPRS.

Proof. This follows from the fact that removing a negative vertex does not change anything in the rest of the graph. Hence, if the reduction $\mathbf{gpr}_P$ of a positive vertex P follows the reduction $\mathbf{gnr}_Q$ of a negative vertex Q, then applying $\mathbf{gpr}_P$ first, followed by $\mathbf{gpr}_Q$will also yield a successful reduction of γ. □

Consequently, in the rest of this section we will only consider canonical reductions.

For 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 $\rho'_1, \ldots, \rho'_l$ is a permutation of $\rho_1, \ldots, \rho_l$.

Lemma 15. Let D_g be a canonical (successful) reduction in GPRS,

$$D_g = (\gamma; \omega_1, \ldots, \omega_l, \rho_1, \ldots, \rho_m),$$

with $\omega_1, \ldots, \omega_l$ positive rules and $\rho_1, \ldots, \rho_m$ negative rules. Then for each permutation $\rho_{i_1}, \ldots, \rho_{i_m}$ of $\rho_1, \ldots, \rho_m$, we have $D'_g = (\gamma; \omega_1, \ldots, \omega_l, \rho_{i_1}, \ldots, \rho_{i_m})$ is also a canonical (successful) reduction.

Proof. Since negative rules apply to isolated vertices only, any two consecutive applications of negative rules can be permuted without changing the result of the whole reduction. □

Remark. Note that Lemma 15 does not hold for SPRS. Indeed, for $\pi = qppq$, the reduction $(\pi; \mathbf{snr}_p, \mathbf{snr}_q)$ is successful, while $(\pi; \mathbf{snr}_q, \mathbf{snr}_p)$ is not even a reduction.

We return now to our goal of proving the translatability of GPRS into SPRS.

Lemma 16. Let π be a legal word and let $p \in \Pi_k$. If $\mathbf{gpr}_P$ is applicable to γ_π, then $\mathbf{spr}_p$ is applicable to π. Moreover, if $\pi' = \mathbf{spr}_p(\pi)$, and $\gamma' = \mathbf{gpr}_P(\gamma_\pi)$, then $\gamma_{\pi'} = \gamma'$.

Proof. If $\mathbf{gpr}_P$ is applicable to P, then the vertex P is positive in γ_π. Consequently, p is a positive pointer in π, and $\mathbf{spr}_p$ is applicable to π. By Lemma 11, we conclude that $\gamma_{\pi'} = \gamma'$. □

From Lemma 16, we obtain by transitivity the following result for positive reduction rules.

Corollary 17. Let π be a legal word. If $(\gamma_\pi; \mathbf{gpr}_{P_1}, \ldots, \mathbf{gpr}_{P_l})$ is a reduction of γ_π in GPRS, then, for some $p_1 \in P_1, \ldots, p_l \in P_l$, we have $(\pi; \mathbf{spr}_{p_1}, \ldots, \mathbf{spr}_{p_l})$ is a reduction of π in SPRS. Moreover, if $\gamma' = \mathbf{gpr}_{P_l} \ldots \mathbf{gpr}_{P_1}(\gamma_\pi)$ and $\pi' = \mathbf{spr}_{p_l} \ldots \mathbf{spr}_{p_1}(\pi)$, then $\gamma' = \gamma_{\pi'}$.

We move now to consider the negative reduction rules.

Lemma 18. Let π be a legal word. If $D' = (\gamma_\pi; \mathbf{gnr}_{P_1}, \ldots, \mathbf{gnr}_{P_l})$ is a successful reduction in GPRS, then there exists a successful reduction $D = (\pi; \mathbf{snr}_{q_1}, \ldots, \mathbf{snr}_{q_l})$ of π in SPRS such

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

that $D'' = (\gamma_\pi, \mathbf{gnr}_{Q_1}, \ldots, \mathbf{gnr}_{Q_l})$ is a successful reduction in GPRS and $\mathbf{gnr}_{Q_1}, \ldots, \mathbf{gnr}_{Q_l}$ is a permutation of $\mathbf{gnr}_{P_1}, \ldots, \mathbf{gnr}_{P_l}$.

Proof. We prove the claim by induction on the length l of the reduction D'.

If $l = 1$, then π must be of the form pp, where $p \in P$, and thus, $D = (\pi; \mathbf{snr}_p)$ is a successful string reduction. Moreover, in this case $D'' = D' = (\gamma_\pi; \mathbf{gnr}_P)$.

Assume now that the claim holds for all reductions of length $n \geqslant 1$, and let $l = n + 1$. Since D' uses only negative reduction rules, γ_π consists of negative isolated vertices only. Hence, in π no two pointers overlap, that is, the pointers in π represent a nested parenthesis structure. Let q be a pointer of nesting level 1, that is, q is such that qq is a subword of π. Hence, the rule $\mathbf{snr}_q$ is applicable to π. Let D'_q be the reduction obtained from D' by moving the reduction $\mathbf{gnr}_Q$ to the front of D'. By Lemma 15, D'_q is a successful graph reduction.

But the rest of D'_q (starting with the second reduction rule) is of length n. Thus, by the induction hypothesis, the claim holds for the rest of D'_q, so the claim holds also for D'. □

Corollary 19. Let π be a legal word. If γ_π is successful in GPRS, then π is successful in SPRS.

Proof. If γ_π is successful in GPRS, then, by Lemma 14, there exists a canonical successful reduction D of γ_π in GPRS. Hence,

$$D = (\gamma_\pi; \mathbf{gpr}_{P_1}, \ldots, \mathbf{gpr}_{P_i}, \mathbf{gnr}_{Q_1}, \ldots, \mathbf{gnr}_{Q_j}),$$

for some pointer sets $P_1, \ldots, P_i, Q_1, \ldots, Q_j$. By Corollary 17,

$$(\pi; \mathbf{spr}_{p_1}, \ldots, \mathbf{spr}_{p_i})$$

is a reduction in SPRS, for some $p_1 \in P_1, \ldots, p_i \in P_i$. Moreover, for

$$\gamma' = \mathbf{gpr}_{P_i} \ldots \mathbf{gpr}_{P_1}(\gamma_\pi) \quad \text{and} \quad \pi' = \mathbf{spr}_{p_i} \ldots \mathbf{spr}_{p_1}(\pi)$$

we have $\gamma' = \gamma_{\pi'}$. Then, by Lemma 18, there exists in SPRS a successful string reduction $(\pi'; \mathbf{snr}_{q'_1}, \ldots, \mathbf{snr}_{q'_j})$. □

The results of this section can be summarized as follows.

Theorem 20. Let π be a legal word. Then π is successful in SPRS if and only if γ_π is successful in GPRS.

This, together with Theorem 9, gives the following result.

Theorem 21. Let δ be a realistic MDS descriptor. The following three conditions are equivalent:

(i) δ has a successful strategy based on $\{\mathbf{ld}, \mathbf{hi}\}$.
(ii) π_δ is successful in SPRS.
(iii) γ_{π_δ} is successful in GPRS.

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

7. Discussion

In this paper we have investigated some formal aspects of gene assembly in ciliates. In particular, we have presented three different formalizations of the structure of a (micronuclear, intermediate or macronuclear) gene: MDS descriptors, legal strings and legal graphs, and then considered the rewriting systems operating on these structures. We have proved that legal graphs, the most abstract of the three structures, contain enough information for the investigation of possible strategies for the gene assembly. Clearly, one would like to isolate those strategies that are most feasible from the biological point of view. This may require more empirical work – as a matter of fact, we have started experiments in this direction. Once we obtain some biological criteria, we can formalize the class of strategies satisfying them, and then develop some techniques for analyzing the corresponding string and graph reduction systems. Graph reduction systems seem to be more 'attractive' for such an investigation, simply because they are 'more abstract'. It seems that these kinds of reduction systems are new, although there is some resemblance to formal systems parsing 'correct parenthesis expressions'. Our reduction systems also constitute new models within the area of DNA computing.

We believe that this paper demonstrates yet another area of applicability of graph transformations, this time to the investigation of *real life* problems.We have merely scratched the surface of the problem area. There is still much to be investigated: for example, the modelling of other operations postulated for gene assembly (see, for example, Ehrenfeucht *et al.* (2000) and Ehrenfeucht *et al.* (2001)), or the characterization of those MDS structures that are admissible as the gene structures in the micronucleus. This latter problem requires both empirical and theoretical investigations (see, for example, the discussion in Prescott *et al.* (2001) concerning simple operations). We are currently working on these problems, and hope to be able to report the results in the future papers.

The computational nature of the gene assembly process has been brought to the attention of the DNA computing community in a series of papers by L.Landweber and L.Kari, see, for example, Landweber and Kari (1998) and Landweber and Kari (1999). However, their model is *intermolecular*, that is, interactions between different molecules form an important part of their model. This is a major difference from our model, which is *intramolecular*, that is, the assembly process is always confined to one molecule (although an IES is excised by the **ld** operation, which is actually common to their model and ours). Moreover, we investigate the assembly process accomplished by various strategies in our reduction systems, while Landweber and Kari investigate the computational power (in the sense of computability theory) of their operations. We believe that their work and ours together shed light on the computational nature of gene assembly in ciliates.

Acknowledgments

This work is supported by NSF Research Grant #MCB-9974680 to David M. Prescott. Ion Petre and Grzegorz Rozenberg gratefully acknowledge support by TMR Network GETGRATS. The authors are grateful to Tero Harju and the anonymous referees for useful comments.

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press

References

Adleman, L. (1994) Molecular computation of solutions to combinatorial problems. *Science* **266** 1021–1024.

Condon, A. and Rozenberg, G. (eds.) (2000) Proceedings of the Sixth International Meeting on DNA Based Computers. *Springer-Verlag Lecture Notes in Computer Science*.

Ehrenfeucht, A., Prescott, D.M. and Rozenberg, G. (2000) Computational aspects of gene (un)scrambling in ciliates. In: Landwerber, L. and Winfree, E. (eds.) *Evolution as Computation*, Springer-Verlag 45–86.

Ehrenfeucht, A., Petre, I., Prescott, D.M. and Rozenberg, G. (2001) Universal and simple operations for gene assembly in ciliates. In: Martin-Vide, C. and Mitrana, V. (eds.) *Words, Sequences, Languages: Where computer science, biology and linguistics come across* 329–342.

Harary, F. (1969) *Graph Theory*, Addison-Wesley.

Landweber, L.F. and Kari, L. (1998) 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.

Landweber, L.F. and Kari, L. (1999) Universal molecular computation in ciliates. In: Landweber, L. and Winfree, E. (eds.) *Evolution as Computation*, Springer-Verlag (to appear).

Păun, G., Rozenberg, G. and Salomaa, A. (1998) *DNA Computing*, Springer-Verlag.

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

Prescott, D.M. (1992b) The unusual organization and processing of genomic DNA in hypotrichous ciliates. *Trends in Genetics* **8** 439–445.

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

https://doi.org/10.1017/S0960129501003516 Published online by Cambridge University Press