Models of natural computation : gene assembly and membrane systems

Brijder, R.

Chapter 1

Introduction

The two main topics of this thesis, gene assembly in ciliates and membrane computing, are representatives of the broad research field of natural computing. Membrane computing is a computational model inspired by the functioning of membranes in living cells, and gene assembly is a complex biological process occurring in unicellular organisms called ciliates.

1.1 Natural Computing

Natural computing is a broad and diverse research discipline residing on the boundary of computer science and natural sciences. Therefore, by its very nature, natural computing is interdisciplinary and it builds bridges between computer science and natural sciences – here computer science is meant as a broadly understood science of information processing. In natural computing one can distinguish two main research directions. On one hand, it considers processes taking place in nature as (some sort of) computation, while on the other hand it is concerned with developing and analyzing computational methods inspired by nature [16].

This thesis considers two research areas within natural computing: gene assembly in ciliates, representing the first research direction given above, and membrane computing, representing the second research direction.

In the following section we recall some very basic cell biology underlying the theory presented in this thesis. Then, in Section 1.3 we provide a basic description of the gene assembly process, and in Section 1.4 we discuss sorting by reversal which is strongly related to our theoretical model of gene assembly. In Section 1.5 we discuss the generic membrane computing model. We conclude this chapter with an outline of the thesis.

1.2 Background: Cells

Each organism consists of one or more cells. The tiniest organisms are unicellular, they consist of just one cell, while, e.g., the number of cells in humans is of the order 10^{14}. On one hand cells can be seen as building blocks for complex organisms such as human beings – cells in such organisms have their own function, and together they form more complex organizations such as tissues, organs, etc. On the other hand, cells themselves are amazingly complex – they have an involved internal structure. Two substructures of (eukaryotic) cells will be most relevant for us: cell membranes and the cell nucleus. A standard text concerning the molecular biology of the cell is [1]. A more accessible text for a computer scientist is Chapter 1 of [10].

1.2.1 Membranes

Membranes separate cells from their environment, but membranes also divide a cell into compartments. Each compartment may have its own structure and function, and either requires or avoids the presence of certain ions and molecules. Communication between compartments of cells or between a cell and its outside environment is facilitated by various kinds of channels. They allow for controlled passage of ions and molecules from one compartment to another, or between the cell and its outside environment. Different channels may control the passage of different molecules or ions.

1.2.2 DNA and Cell Nucleus

A single-stranded DNA molecule (where DNA stands for deoxyribonucleic acid) is a chain of basic components (monomers) called nucleotides. There are four types of nucleotides: adenine, cytosine, guanine, and thymine, abbreviated as A, C, G, and T, respectively. A DNA molecule can thus be represented as a sequence of symbols A, C, G, and T. For example, the sequence (string) $GACGT$ represents a single-stranded DNA molecule, which is the chain of nucleotides G, A, C, G, T (in this order).

A single stranded DNA molecule has a natural orientation, meaning that one end of it is (chemically) distinguishable from the other – one of the ends is called 5' and the other one 3'. Almost all information processing of DNA molecules in nature happens in the direction from 5' to 3', and for this reason the reading of the sequence of the nucleotides comprising a DNA molecule goes from its 5' end to its 3' end. Hence, DNA molecule $GACGT$ is *not* equal to its reverse $TGCAG$.

A basic feature of single stranded DNA molecules is that each such molecule has a complementary single stranded DNA molecule. Together they can form a double stranded DNA molecule. Here, two complementary single stranded DNA molecules bind together by weak hydrogen bonds between complementary nucleotides: nucleotides A and T are complementary, and C and G are complemen-

Figure 1.1: Two single-stranded DNA molecules forming a double-stranded DNA molecule.

tary. Moreover, the two complementary single stranded DNA molecules bind in their opposite orientation – meaning that the first (second, .., resp.) nucleotide on the 5' end of one molecule sticks to the first (second, .., resp.) nucleotide on the 3' end of the other molecule. This is illustrated in Figure 1.1 with the complementary DNA molecules $GACGT$ and $ACGTC$. We use the arrows as the standard notation for indicating the 5'-3' orientation of a single-stranded DNA molecule. Since strings are used to denote/specify single-stranded DNA molecules, the double string notation is very natural for denoting double-stranded DNA molecules. Thus, the double-stranded DNA molecule in the Figure 1.1 is denoted by either $\begin{matrix} GACGT \\ CTGCA \end{matrix}$ or $\begin{matrix} ACGTC \\ TGCAG \end{matrix}$, since double-stranded DNA molecules do not have an orientation. Of course, segments within a double-stranded DNA molecule α do have an orientation (w.r.t. α), e.g., although double-stranded DNA molecule $\begin{matrix} AC \\ TG \end{matrix}$ can also be represented as $\begin{matrix} GT \\ CA \end{matrix}$, only $\begin{matrix} AC \\ TG \end{matrix}$ appears in $\begin{matrix} GACGT \\ CTGCA \end{matrix}$ which is *not* equal to $\begin{matrix} GGTGT \\ CCACA \end{matrix}$. For this reason, we sometimes fix an orientation of a double-stranded DNA molecule, i.e., choose one of the two representations of the molecule. If we let M be a double-stranded DNA molecule with a fixed orientation, then we define the *inversion* of M, denoted by $\bar{M}$, to be the same double-stranded DNA molecule with the other orientation, i.e., M rotated 180 degrees.

The cell nucleus is a substructure of the cell holding the genome. The genome is divided into a number of chromosomes, e.g., the human genome consists of 46 chromosomes. Each chromosome contains one double-stranded DNA molecule. These DNA molecules contain genes which are segments containing "instructions" for the production (expression) of proteins. The genetic part of chromosomal DNA may be very small (e.g., in humans only about 2%-5% is genetic). It also contains regulatory information (when and how much of specific proteins should be produced), but the role of the non-genetic part of chromosomal DNA is not yet well understood.

1.3 Gene Assembly in Ciliates

Ciliates (ciliated protozoa) are a group of ancient unicellular organisms. The name ciliates is due to the hair-like structure, called cilia, present on their external surface. Ciliates are different from other organisms in that they have two kinds of

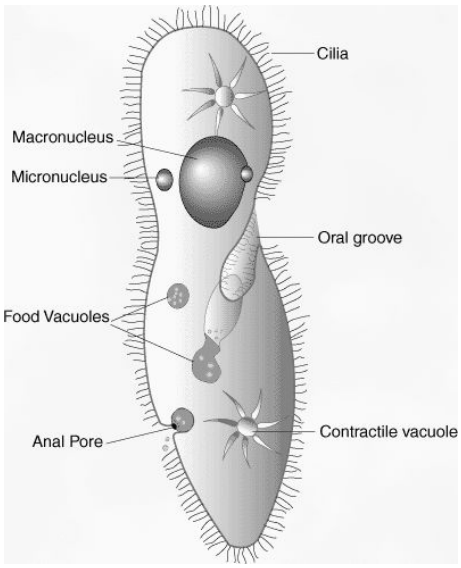

Figure 1.2: Schematic image of a ciliate, copyright SparkNotes.

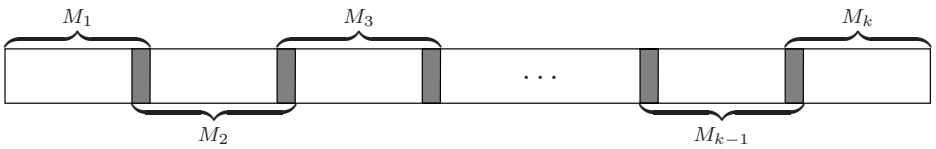

Figure 1.3: The structure of a MAC gene consisting of κ MDSs.

nuclei that are radically different, both functionally and physically. The two kinds of nuclei (which both can be present in various multiplicities) are called micronucleus (MIC) and macronucleus (MAC) – the former is used only in mating, while the latter is used for producing RNA needed for cell maintenance and reproduction. A schematic image of a ciliate is given in Figure 1.2.

The number of chromosomes in a MIC is similar to that of other eukaryotes (say about 100), while there are very many (millions) of minichromosomes in the MAC. Also, the MIC chromosomes are very long (as is generally the case for eukaryotes) and for the most part (more than 95%) non-genetic. The MAC minichromosomes are very short (on average about 2000 base pairs but may be as small as 300 base pairs), and for the most part (about 85%) genetic.

All the genes occur in both the MIC and the MAC, but in very different forms. The relationship between the form of MIC and MAC genes can be described as follows. For each gene, one can distinguish a number of double stranded DNA molecules $M_1, \ldots, M_\kappa$ with a fixed orientation, called MDSs (macronuclear destined

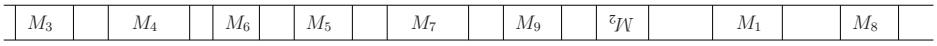

Figure 1.4: The structure of the MIC gene encoding for the actin protein in sterkiella nova.

segments), appearing in both the MIC and MAC form of that gene. The MAC form is a sequence of overlapping MDSs in their orthodox order, i.e., $M_1, \ldots, M_\kappa$ – this is illustrated in Figure 1.3. The gray areas in the figure indicate the overlaps of MDSs – these overlaps are called pointers. In the MIC form the MDSs are separated by non-coding segments, called IESs (internal eliminated segments). The MDSs either occur in orthodox order or in a different order, and the MDSs can occur inverted (no MDS can occur twice). As an example, Figure 1.4 shows the MIC form of the gene that encodes for the actin protein in a ciliate called sterkiella nova. This gene consists of nine segments, where the enumeration $M_1, M_2, \ldots, M_9$ refers to the orthodox order of the MDSs in the MAC form of the gene. Note that MDS M_2 occurs inverted in the MIC form of the gene. It is important to realize that the number of MDSs, the specific permutation of the MDSs and the possible inversions are fixed for a given gene and given species, but they can be very different for different genes and for the same gene in different species.

The process of gene assembly transforms a MIC into a MAC. This process occurs during sexual reproduction of two ciliates where first a MIC is formed holding half of the genetic information of each parent, and then a MAC is constructed from this newly formed MIC. During gene assembly, each of the about 25,000 genes in MIC form are transformed into the corresponding gene in MAC form. The transformation of a single gene from MIC form to MAC form is complex: all MDSs must be 'sorted' in the right order and must have the right orientation, and all IESs must be spliced out from between the MDSs. This transformation process involves quite a number of "cutting and gluing" of DNA. Pointers in the MIC form indicate how this cutting and gluing, called recombination, is done. Indeed, each overlapping segment of two MDSs in the MAC form appears in two places in the MIC form and this in turn indicates where the DNA segments are to be cut and glued together. The differences in the genetic material of the MIC and the MAC discussed above are particularly pronounced in the stichotrichs group of ciliates. For this reason, a lot of literature, including this thesis, concerns this group of ciliates. We refer to [10] for an in-depth treatment of the biology of gene assembly.

In one possible modeling of the assembly process, the MIC form of a gene is transformed into the MAC form through three types of recombination operations that operate on the pointers. These types of operations are called: loop recombination, hairpin recombination, and double-loop recombination. Each of these recombinations can only take place on pointers of the gene in MIC form (or an intermediate product) provided that these pointers fulfill specific conditions. The

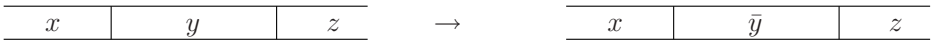

Figure 1.5: Inversion within a chromosome.

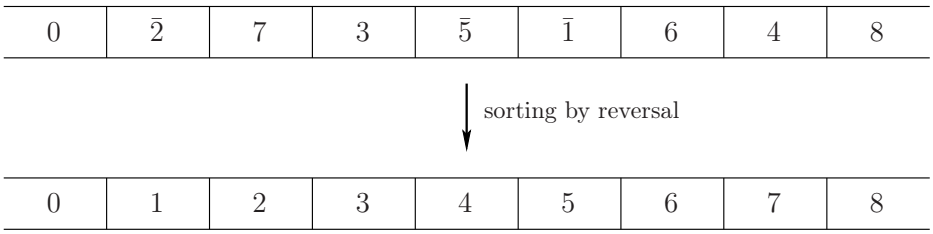

Figure 1.6: Two chromosomes of different species and their common contiguous segments.

operations are defined in [10] and we will recall them in Part 1 of this thesis.

1.4 Sorting by Reversal

During evolution the genomes of species change. One such change is inversion, and is illustrated in Figure 1.5. The result is that a segment y is inverted (rotated 180 degrees) – this is indicated by $\bar{y}$ in the figure. In this way, two different species can have several contiguous segments in their genome that are very similar, although their relative order (and orientation) may differ in both genomes. For example, consider the two chromosomes in Figure 1.6. Both chromosomes have 9 segments in common, however their relative order and orientation differs. The breakpoints of a chromosome are the borders of each two consecutive segments. Figure 1.7 shows the application of an inversion, called reversal, on the breakpoint between segments 0 and $\bar{2}$ and the breakpoint between segments $\bar{1}$ and 6 (these two breakpoints are indicated by two small arrows in the figure).

In the theory of sorting by reversal, initiated by S. Hannenhalli and P.A. Pevzner in [11], one tries to determine the minimal number of reversals needed to convert the genome of one species into that of the other. The smaller this number, the more likely it is that their common ancestor is relatively young in evolution. Thus, this number can aid in constructing an ancestor tree of species, called a phylogenetical tree.

Note that sorting by reversal differs from gene assembly in ciliates in several aspects. First, it is an evolutionary process from one species to another; there are no pointers that indicate where recombination should take place. Second, recombination takes place on the scale of complete chromosomes, while in gene

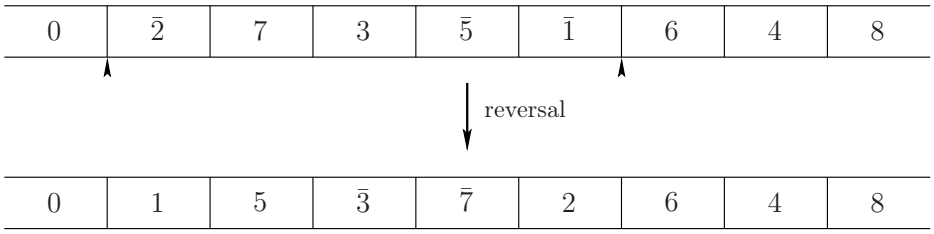

Figure 1.7: Applying a reversal on the chromosome.

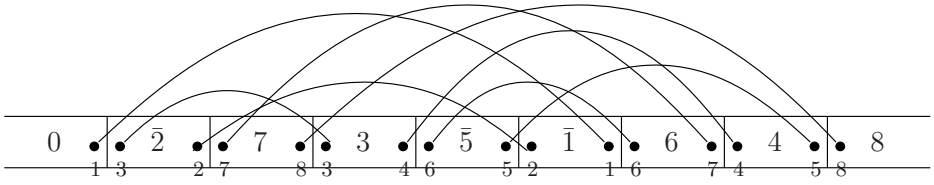

Figure 1.8: The breakpoint graph of the given chromosome.

assembly it is on the level of individual genes. And finally, instead of three types of recombination operations there is only one type: the reversal.

An essential tool in the theory of sorting by reversal is the breakpoint graph (also called reality and desire diagram) which is used to capture both the present situation, the genome of the first species, and the desired situation, the genome of the second species. For each breakpoint, we assign two vertices in the graph representing both sides of that breakpoint. These vertices are labeled such that segment i has vertices labelled by i and $i+1$. Then i represents the left-hand side and $i+1$ the right-hand side of segment i. If segment i appears inverted in the genome then, w.r.t. the chromosome, i appears on the right-hand side and $i+1$ on the left-hand side. Moreover, there are edges, called desire edges, that connect vertices with the same label. In Figure 1.8 these vertices and edges are depicted for our example*.

In addition to the desire edges, the breakpoint graph has a second set of edges, called reality edges. These edges connect each two vertices belonging to the same breakpoint. Thus, in Figure 1.8, the left-most two vertices labeled by 1 and 3 are connected by a reality edge, and similarly for the next two vertices labeled by 2 and 7, etc. The linear order of the vertices in the figure is therefore partially captured by the reality edges. However, the complete linear order of the vertices remains important, and therefore the breakpoint graph should not be seen as a graph, but

*It is customary for breakpoint graphs to let $2i-1$ represent the left-hand side and $2i$ the right-hand side of segment i – in this way eliminating the need for labels. However, we choose this notation to make comparison with reduction graphs (defined in the next chapter) easier.

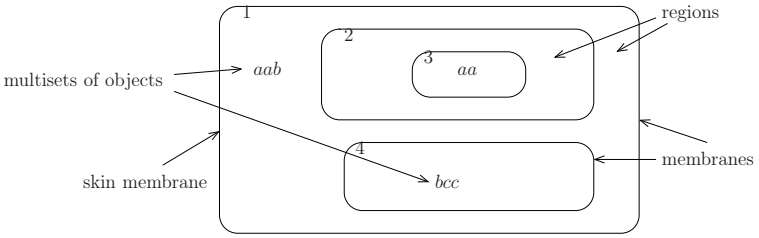

Figure 1.9: Example membrane system.

as a diagram where the vertices are drawn in this linear order. Therefore reality and desire *diagram* is arguably a more appropriate name for this concept. One could extent the breakpoint graph with a third set of edges, for example called segment edges, connecting each two consecutive vertices belonging to the same segment. Thus, e.g., in Figure 1.8 the two vertices labeled by 3 and 2 of segment $\bar{2}$ are then connected by such a segment edge. In this way, we obtain a graph which retains the linear order of the vertices, and hence need not be seen as a diagram. We will introduce these additional sets of edges in the context of gene assembly in this thesis. Given only the breakpoint graph it is possible to deduce, in a computationally efficient way, the minimal number of reversals needed to convert the genome from one species into that of the other.

1.5 Membrane Computing

Membrane computing studies a range of computational models inspired by the functioning of membranes in cells. This research area was initiated by Gh. Păun in 1998 (see [13]). Membrane systems are therefore also often called P systems after its inventor. Membrane computing has in a short time attracted a large research community. Many classes of membrane systems exist, but in this section we consider a 'typical/generic' membrane system; for an in-depth introduction to membrane computing we refer to [14], and for an easier-to-read overview we refer to [15].

Such a membrane system consists of a hierarchical membrane structure where each membrane, except for the outer membrane (called the skin membrane), is fully contained in another membrane (called its parent). The compartments enclosed by (situated in-between) the membranes are called regions. An example of a membrane structure is given in Figure 1.9.

Each region contains zero or more objects, and each object is of a certain type. In the figure the region enclosed by the skin membrane (called skin region) contains two objects of type a, and one object of type b. To make the system evolve/compute there are evolution rules assigned to the regions that in some way transform, create, delete, or move the objects (between regions – moving

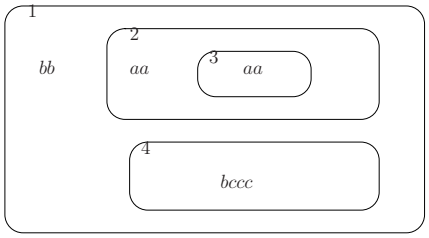

Figure 1.10: A possible state of the membrane system in Figure 1.9 after one time step.

objects between adjacent regions is referred to as communication).

In each time step (there is a global clock) during the evolution of a membrane system many such evolution rules can be applied in parallel. In fact, in each time step the evolution rules are applied in a *maximal* parallel manner: the multiset of evolution rules that is applied cannot be extended by any evolution rule – no subset of the objects that remain unused in a given time step can evolve using any evolution rule.

For example, there could be two evolution rules in the skin region: one that transforms one object of type a and one of type b into two objects of type a, both of which cross membrane 2, *and* one that transforms one object of type a into two objects of type b (both staying in the skin region). In addition there could be an evolution rule in the region enclosed by membrane 4 transforming one object of type b and one of type c into one object of type b and two objects of type c. Then, there are two possible maximal parallel ways to transform this membrane system. In the next time step, the state of the membrane system is either the one given in Figure 1.10 or the one given in Figure 1.11.

A membrane system computes by iteratively applying the evolution rules in a maximal parallel manner until no evolution rule can be applied anymore. Then, the contents of a preselected membrane, called the output membrane, is the result of the computation. In this way, the language of a given membrane system is defined to be the set of results of all computations of the membrane system.

A well-studied class of membrane systems called symport/antiport P systems involves only communication and no transformation, see [12]. Here, the rules are assigned to membranes instead of regions, and they allow movement of objects from a region on one side of the membrane to the region on the other side. The movement of objects is also synchronized. For example, an object a may only move together with object b to the other side of the membrane. Or, for example, an object a may only move through the membrane if simultaneously an object b from the other side of the membrane moves through the membrane in the opposite direction. The former type of movement is described by the so-called symport rules, while the latter type of movement is described by antiport rules. Note that

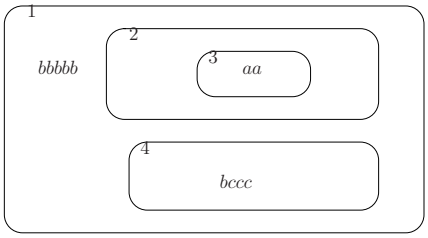

Figure 1.11: A possible state of the membrane system in Figure 1.9 after one time step.

these rules cannot change either the type of an individual object or the quantity of objects present in the system.

1.6 Overview of the Thesis

This thesis consists of two parts.

The first part, consisting of Chapters 2 through 5, is devoted to gene assembly in ciliates. The central notion of this part is the reduction graph – it is inspired by the breakpoint graph discussed in Section 1.4. The concept of reality and desire remains in place, but the notion of reduction graph is specifically tailored for the theory of gene assembly. Given the MIC form (reality) of a gene, the reduction graph describes the end result (desire) after gene assembly for this gene is completed. This includes the MAC form of a gene, but it also describes the "end structure" of all the IESs. The reduction graph however is defined in a more general fashion: it deals with arbitrary recombination using pointers. In the model we use, the MIC form of the gene is represented by a string, called legal string, and the reduction graph is defined for each such legal string. In Chapter 2 we introduce the reduction graph, and then use it to characterize the intermediate gene patterns that may occur during the transformation of a MIC form of a gene to its MAC form. We also show that for legal strings in general the number of loop recombination operations in each possible strategy transforming the MIC form into the MAC form is fixed and directly determinable through the reduction graph. This chapter is based on [7]. In Chapter 3 we strengthen these results in order to obtain a characterization of loop recombination that allows one to determine which loop recombination operations can be applied in such strategies and also in which order they can be applied. This is done by using the notion of pointer-component graph (defined "on top of" the reduction graph) that identifies the relationship of pointers on the connected components of the reduction graph. This chapter is based on [6]. Since the reduction graph is the main notion of the first part of the thesis, it is certainly natural to ask which graphs are reduction graphs. Such a characterization of reduction graphs is given in Chapter 4. Also,

in Chapter 4 we consider the problem of equivalence for MIC genes: we characterize which genes in MIC form (formally legal strings) yield the same end result after gene assembly is accomplished. This characterization is given in terms of string rewriting rules (applied to legal strings) that correspond to recombination operations which, surprisingly, are very similar to the recombination operations defining gene assembly. This chapter is based on [5]. The MIC forms of genes can be represented both as strings, called legal strings, and as graphs, called signed overlap graphs. Both representations lead to two almost equivalent models of gene assembly. The definition of reduction graph in Chapter 2 relies on string representations. In Chapter 5 we define the reduction graph directly for overlap graphs, and show that this graph is identical to the reduction graph of every "realistic" legal string corresponding to that overlap graph. This allows one to carry over the results of Chapters 2, 3, and 4 to the graph based model of gene assembly. This chapter is based on [9] (see [8] for an extended abstract).

The second part of this thesis, consisting of Chapters 6 to 8, is devoted to membrane computing. Chapter 6 considers membrane systems that can have objects not only within the regions (which is standard in membrane systems) but also on/within the membranes themselves. Such objects allow for both controlled movement of objects through the membranes and controlled evolution of the membranes. These systems are biologically motivated by the fact that some proteins (represented by objects) residing on/within membranes control the movement of ions/molecules through membranes. This chapter is based on [4]. Chapter 7 considers membrane systems where the evolution of the system depends on external signals. Each signal is represented by a sequence (string) of objects which enters the system from outside and during the evolution of the system moves through the regions. Here the first object of the signal has influence on the system and this object is removed when passing through a membrane until finally the whole "string signal" has disappeared. This chapter is based on [3]. Chapter 8 focusses on membrane systems with symports and antiports where we relax the condition that symports only move objects – we allow now that during the crossing of a membrane the objects themselves can change in both type and quantity. The intuitive interpretation is that objects can engage in (biochemical) reactions while crossing the membrane. This chapter is based on [2]. The central unifying research topic (question) which we consider in Part 2 is the computational power (including decidability results) of the various classes of membrane systems described above.

Bibliography

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *The Molecular Biology of the Cell.* Garland Publ. Inc., London, 4th edition, 2002.

[2] R. Brijder, M. Cavaliere, A. Riscos-Núñez, G. Rozenberg, and D. Sburlan. Communication membrane systems with active symports. *Journal of Automata, Languages and Combinatorics*, 11(3):241–261, 2006.

[3] R. Brijder, M. Cavaliere, A. Riscos-Núñez, G. Rozenberg, and D. Sburlan. Membrane systems with external control. In H.J. Hoogeboom, G. Paun, G. Rozenberg, and A. Salomaa, editors, *Workshop on Membrane Computing*, volume 4361 of *Lecture Notes in Computer Science*, pages 215–232. Springer, 2006.

[4] R. Brijder, M. Cavaliere, A. Riscos-Núñez, G. Rozenberg, and D. Sburlan. Membrane systems with proteins embedded in membranes. *Theoretical Computer Science*, 404:26–39, 2008.

[5] R. Brijder and H.J. Hoogeboom. The fibers and range of reduction graphs in ciliates. *Acta Informatica*, 45:383–402, 2008.

[6] R. Brijder, H.J. Hoogeboom, and M. Muskulus. Strategies of loop recombination in ciliates. *Discrete Applied Mathematics*, 156:1736–1753, 2008.

[7] R. Brijder, H.J. Hoogeboom, and G. Rozenberg. Reducibility of gene patterns in ciliates using the breakpoint graph. *Theoretical Computer Science*, 356:26–45, 2006.

[8] R. Brijder, H.J. Hoogeboom, and G. Rozenberg. From micro to macro: How the overlap graph determines the reduction graph in ciliates. In E. Csuhaj-Varjú and Z. Ésik, editors, *Fundamentals of Computation Theory (FCT) 2007*, volume 4639 of *Lecture Notes in Computer Science*, pages 149–160. Springer, 2007.

[9] R. Brijder, H.J. Hoogeboom, and G. Rozenberg. How overlap determines the macronuclear genes in ciliates. Submitted, also LIACS Technical Report 2007-02, [arXiv:cs.LO/0702171], 2008.

[10] A. Ehrenfeucht, T. Harju, I. Petre, D.M. Prescott, and G. Rozenberg. *Computation in Living Cells – Gene Assembly in Ciliates*. Springer Verlag, 2004.

[11] S. Hannenhalli and P.A. Pevzner. Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *J. ACM*, 46(1):1–27, 1999.

[12] A. Păun and Gh. Păun. The power of communication: P systems with symport/antiport. *New Generation Computing*, 20(3):295–305, 2002.

[13] Gh. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000. Also, Turku Center for Computer Science-TUCS Report No. 208, 1998.

[14] Gh. Păun. *Membrane Computing. An Introduction*. Springer, Berlin, 2002.

[15] Gh. Păun and G. Rozenberg. A guide to membrane computing. *Theoretical Computer Science*, 287(1):73–100, 2002.

[16] G. Rozenberg. Computer science, informatics, and natural computing - personal reflections. In S.B. Cooper, B. Löwe, and A. Sorbi, editors, *New Computational Paradigms - Changing Conceptions of What is Computable*, pages 373–379. Springer, 2007.