Aqueous Simulations of Membrane Computations

Tom Head
Mathematical Sciences, Binghamton University
Binghamton, New York 13902–6000

Abstract   A scheme of definitions that formalizes a limited family of membrane computations is presented. The purpose is to provide a link between one of the approaches to biomolecular computation, called aqueous computing, and the newly developing concept of membrane computing, also called P–systems computing. This link allows one to view the already completed aqueous computations as wet lab realizations, or test tube simulations, of P–systems computations. It is hoped that the elementary linkage established here will be expanded and that it will suggest further developments in both P–systems and aqueous computing.

Key words: membrane computing, P–systems, aqueous computing, molecular computing, DNA computing

1. Introduction.

Within the general context of *natural computing*, as understood and encouraged through the Leiden Center for Natural Computing (LCNC), two of the *unconventional models of computation* that have been developed are *aqueous computing* and *membrane computing*. Aqueous computing was initiated as a specific pattern of wet lab DNA computing. The concept was spelled out in a preliminary form in [H'00]; fleshed out in laboratories as reported in [HRBBLS'00], [HYG'99] and [YHG'00]; and described in [H'01a]. Further laboratory computations are now in progress in the aqueous manner. Membrane computing was introduced by Gheorghe Paun [P'00] as a multifaceted model of computation. An international research community has developed to investigate these new *membrane systems,* which are now also called *P–systems*. Membrane systems appear already in the book by C. Calude & Gh. Paun [CP'01]. Valuable additional background references for this general area are [P'98], [PRS'98], [CCD'98] and [ACD'01]. Two relevant and valuable bibliographic web pages are being maintained, one by Pierluigi Frisco on DNA computing [Fweb] and one by Claudio Zandron on P–systems [Zweb].

The purpose of the present article is to link the wet lab based aqueous computing with the theory of P–systems. The P–system theoretical results presented in [ACD'01] suggested that wet lab computations realizing (or simulating) appropriate membrane computations should be possible. In fact, the three aqueous computations that have already been completed in Leiden and in Binghamton can be regarded as wet lab realizations of certain simple forms of membrane computations. We develop these specific membranes systems here as re–write systems. These systems provide a representation of not only the three aqueous computations that have been completed, but also of a fourth that is in progress and a fifth that has been planned in detail.

The rules for the membrane formalism used here are carefully illustrated in Section 2. A derivation using the first five of these rules is given in Section 3 as an illustration of the application of the membrane formalism to the solution of the problem of finding the maximal independent sets of vertices of a graph. In Section 4 wet lab procedures of aqueous computing are correlated with applications of the rules presented in Section 2 to show that an already completed aqueous computation [HRBBLS'00] can be interpreted as a realization of the derivation given in Section 3. A final paragraph of Section 4 completes the correlation between the derivation rules and their aqueous realizations. This allows a previously completed aqueous computation [HYG'99] [YHG'00] to be interpreted as a realization of the derivation given in Section 5 which illustrates the application of the membrane formalism to the solution of satisfiability problems (SAT). Section 6 provides a simplified summary of the correlation established here.

2. A Membrane Formalism Conveniently Related to Aqueous Computation.

Only three basic symbols for membranes will be required here. SK will be the unique, permanent, outermost membrane, called *the skin*. Membranes other than SK will be located within SK and be called *proper* membranes. Membranes that do not contain other membranes will be called *simple*

membranes. M will be used to denote proper simple membranes. N will be used to denote proper membranes that, at least initially, are not simple. Each membrane, other than the skin, will at all times be in a specified state. Proper membranes will be denoted in the form: M[StateName] or N[StateName]. A membrane may have, as its *content*, strings over a specified alphabet set, V, and, possibly, other membranes. The content of each membrane is listed inside braces immediately following the name of the membrane. For the illustration in Section 3, V = {a,b,c,d,e,f} and the computation begins with the membrane configuration:

SK{ab, bc, cd, de, M[0]{abcdef}}.

Thus we have inside the skin four strings ab, bc, cd, and de, and also a membrane M[0], in state 0, that contains only one string abcdef. For the illustration in Section 5, V = {A,a,B,b,C,c} and the computation begins with the membrane configuration:

SK{ab, AbC, BC, Ac, N[0]{Aa, Bb, Cc, M[0]{AaBbCc}}}.

Thus we have, inside the skin, the four strings ab, AbC, BC, and Ac, and also a membrane N[0], in state 0, that contains the three strings Aa, Bb, and Cc, and also a membrane M[0], in state 0, that contains only one string AaBbCc.

The actions allowed within membranes are communicated, somewhat informally, by supplying one example of each of several string replacement rules. For this purpose we use the small alphabet **V** = {a,b,c,d}. BAL will denote any sequence of symbols that may occur within the bounding pair of braces of a membranes expression SK{ … } for which each brace in BAL occurs as one member of a pair of matching braces and the usual syntactic relations among the pairs of braces within BAL hold. (The reader may find that it is a helpful mnemonic to think of the membranes as living cells.)

The Proposed Rule Set

(1) bda, M[0] M[1,bda]
Imagery: A membrane M[0] may 'eat' an adjacent string, resulting in a change of state of the membrane.

(2) M[1,bda]{ BAL } M[1,−−a]{a'(BAL)}, M[1,−d−]{d'(BAL)}, M[1,b−−]{b'(BAL)}
(Where a', d' and b' are operation sysmbols applied to the sequence of strings occuring in BAL.)
Imagery: After a string enters a membrane the membrane subdivides into separate membranes, one membrane for each letter in the string. The individual letters act as operators on the content of the resulting membranes.

(3) M[1,−d−]{d'(BAL)} M[2,−d−]{ BAL' }
(Where BAL' is the sequence of strings obtained by **deleting** d **from each string** in BAL.)
Imagery: The completion of the action of an operator within a membrane provides a change of state.

(4) M[2,−−a]{ BAL_1 }, M[2,−d−]{ BAL_2 }, M[2,b−−]{ BAL_3 } M[0] { BAL_1, BAL_2, BAL_3 }
Imagery: A collection of membranes with each in an appropriate state may merge. But observe that, in order to produce a single membrane in state M[0], a certain 'completeness' is required on the set of states of the merging membranes. In the present case all three of −−a, −d−, and b−− must appear in order to yield a membrane in state M[0].

(5) SK{M[0]{ BAL }} SK{ BAL}
Imagery: When no 'food' remains for a membrane M[0] to 'eat', the membrane dissolves, leaving its content in the membrane in which it is immediately contained.

(6) N[0]{M[0]{ BAL }} N[1]{ BAL }
Imagery: When M[0] dissolves inside a membrane N[0], it changes the state to N[1].

(7) bda, N[1] N[2,bda]
Imagery: A membrane N[1], may 'eat' an adjacent string, resulting in a change of state.

(8) N[2,bda]{ BAL } N[2,−−a]{a"(BAL)}, N[2,−d−]{d"(BAL)}, N[2,b−−]{b"(BAL)}
(Where a", d" and b" are operation symbols applied to each of strings occuring in BAL.)

Imagery: After a string enters a membrane the membrane subdivides into separate membranes, one membrane for each letter in the string. The individual letters act as operators on the content of the resulting membranes.

(9) N[2,−d−]{d"(BAL)} N[3,−d−]{ BAL" }
        (Where BAL" is the sequence of strings obtained by **deleting each string** in BAL that contains d.)
Imagery: The completion of the action of an operator within a membrane provides a change of state.

(10) N[3,−−a]{ BAL_1 }, N[3,−d−]{ BAL_2 }, N[3,b−−]{ BAL_3 } N[1] { BAL_1, BAL_2, BAL_3 }
Imagery: Membranes in appropriate states may merge.

(11) SK{N[1]{ BAL }} SK{ BAL}
Imagery: When no 'food' remains for a membrane N[1] to 'eat', the membrane dissolves, leaving its content in the membrane in which it is contained.

        We propose an optional rule of interaction between strings that lie in precisely the same set of membranes. This rule will be used in Section 3. In Section 5 we may consider that either the rule is used or not used, since no setting arises in which it could be applied in any case.

(Optional 12) u, v v is applied *with highest priority* whenever each letter in u occurs also in v.
Imagery: When every symbol in u occurs also in v, u is 'absorbed' into v. Note that since u, u u, we are certainly using sets, rather than multi−sets.

        We regard the content of each membrane as a set. *Thus the order in which the members are listed inside a membrane can be changed at any time.* With this understanding, rules such (1), (7), and (12) may be applied, after reordering the strings, to strings that are not initially listed at the immediate left of the membranes appearing in these rules. Moreover, the order of the three membranes that appear on the right of rules (2) & (8) and on the left of rules (4) & (10) can be reordered at will.
Imagery: The contents of a membrane 'float about freely' within the membrane.
        In Section 3 we confirm that when rules (1) through (5) and also (12) are applied to the membrane SK{ab, bc, cd, de, M[0]{abcdef}} the sequence of applications terminates naturally (and inevitably) in the membrane SK{adf, acef, bef, bdf}. *When a membrane computation results in a Skin that contains only strings, we says that the membrane computation has terminated successfully and that it has computed the set consisting of the strings remaining in SK*. In Section 5 we confirm that the application of rules (1) through (11) to the membrane SK{ab, AbC, BC, Ac, N[0]{Aa, Bb, Cc, M[0]{AaBbCc}}} results in the membrane SK{aBc}. Thus in Sections 3 and 5 the sets computed by the membrane systems are {adf, acef, bef, bdf} and {aBc}, respectively.
        Finally, we remark that the various strings used here may be regarded as notations for sets, (although this is not required). For example, in Section 3, abcdef is merely a short compact notation for the set {a,b,c,d,e,f} and the string ab is likewise an alternate notation for the set {a,b}. We use the string representation for these sets to avoid producing a further level of nesting of braces. Thus we may write abcdef = {a,b,c,d,e,f} = defcba but, as will be seen below, no occasion arises for the use of such equivalent formulations.

3. A Membrane System for the Maximal Independent Set Problem.

        We have reported in [HRBBLS'00] the wet lab aqueous solution of the following instance of the Maximum Independent Set problem [GJ'79]. Let G = (V,E) be the undirected graph having as the vertex set V = {a,b,c,d,e,f} with the four edges {a,b}, {b,c}, {c,d}, {d,e} constituting E. A subset S of V is *independent* if no edge in E is a subset of S. A subset S of V is a *maximal independent set* if it is independent and it is properly contained in no other independent subset. A subset S of V is a *maximum independent subset* if it is independent and no independent subset of V has cardinal number greater than the cardinal number of S. Apparently each maximum independent subset is also a maximal independent subset. For the graph G = (V,E), we represent the problem of finding all the maximal independent sets of G as the initial setting of a membrane computation. This single example indicates how the maximal independent set problem can be treated for any unordered graph.

We place the elements of V, expressed as a string abcdef, in a membrane M[0]. We then place the membrane M[0] and its content into the membrane SK along with the four edges in E, expressed as the strings ab, bc, cd, de. This gives the initial line of the computation immediately below. The computation then proceeds through seventeen applications of the rules in the sequence $(1,2,3,4)^4 5$. The computation terminates and yields the set {adf, acef, bef, bdf}, which is precisely the set of all *maximal independent sets*, i.e., there are four maximal independent sets: {a,d,f}, {a,c,e,f}, {b,e,f}, {b,d,f}. An inspection of the solution of this problem gives also the solution of the associated maximum independent set problem: {a,c,e,f} is the unique *maximum independent set*.

We have applied the optional Rule 12. If it had not been applied, then at the final step a few additional strings would have been included in SK that represent independent sets that are not maximal.

SK{ab, bc, cd, de, M[0]{abcdef}}
    SK{ab, bc, cd, M[1,de]{abcdef}}
    SK{ab, bc, cd, M[1,−e]{e'(abcdef)}, M[1,d−]{d'(abcdef)}}
    SK{ab, bc, cd, M[2,−e]{abcdf}, M[2,d−]{abcef}}
SK{ab, bc, cd, M[0]{abcdf, abcef}}
    SK{ab, bc, M[1,cd]{abcdf, abcef}}
    SK{ab, bc, M[1,−d]{d'(abcdf, abcef)}, M[1,c−]{c'(abcdf, abcef)}}
    SK{ab, bc, M[2,−d]{abcf, abcef}, M[2,c−]{abdf, abef}}
SK{ab, bc, M[0]{abcef, abdf}}
    SK{ab, M[1,bc]{abcef, abdf}}
    SK{ab, M[1,−c]{c'(abcef, abdf)}, M[1,b−]{b'(abcef, abdf)}
    SK{ab, M[2,−c]{abef, abdf}, M[2,b−]{acef, adf}}
SK{ab, M[0]{abef, abdf, acef}}
    SK{M[1,ab]{abef, abdf, acef}}
    SK{M[1,−b]{b'(abef, abdf, acef)}, M[1,a−]{a'(abef, abdf, acef)}}
    SK{M[2,−b]{aef, adf, acef}, M[2,a−]{bef, bdf, cef}}
SK{M[0]{adf, acef, bef, bdf}}
SK{adf, acef, bef, bdf}

4. Viewing Aqueous Computations as Wet Lab Realizations of Membrane Computations.

Each proper simple membrane occurring in a membrane computation is realized in the corresponding aqueous computation as a test tube containing molecules dissolved in water. The *molecules* encode the strings that appear as the content of the membrane. Strings that lie in a non−simple membrane, but outside any simple membrane, specify *processes* that must be carried out during the computation. Thus the first line of the computation in Section 3 indicates that the corresponding aqueous realization begins with a test tube M containing only a single molecular variety that encodes the string abcdef. This first line also indicates that four processes will be carried out on the content of M. Line five indicates the result of carrying out the process specified by the string de. The result of carrying out de gives a test tube M containing two molecular varieties encoded by the strings abcdf and abcef. The intermediate lines 2, 3, and 4 indicate transition states occurring during the following four steps in the aqueous computation:
The application of Rule 1 indicates that the decision has been made to apply the process de to the content of M. Rule 2 indicates that the content of M is poured into two test tubes. In one tube an operation e' (e−deletion) is applied which alters the molecular variety encoding abcdef so that the new molecular variety encodes abcdf. In the other tube an operation d' (d−deletion) is applied which alters the molecular variety so that the new molecular variety encodes abcef. Rule 3 indicates the completion of a deletion process. Rule 4 indicates that the two tubes are returned to a single tube M[0] which is now in the appropriate state to allow another similar process to be applied. Three additional four step cycles, corresponding to cd, bc, ab, are carried out in precisely the same manner. Then Rule 5 applies and recognizes that the computation is complete and that the final tube, now denoted by SK, contains precisely those molecular varieties that encode the set of strings representing the maximal independent sets of vertices of the graph.

The aqueous computation that is interpreted here, as a realization of a membrane system computation, has already been published [HRBBLS'00]. It was carried out using circular DNA molecules (plasmids) to encode the strings that appear inside the membranes M. The operations, e', d',

c', b', a' were realized as compound biochemical steps involving restriction enzymes, a ligase enzyme, bacterial amplification, and purifying procedures. The reader who wishes to know the details of the biochemistry involved in the aqueous computation should begin with [HRBBLS'00]. For the future we hope that aqueous computing can be done rapidly by replacing the biochemical technology with laser technology [H'01a]. *Whatever technology is used, aqueous computations will continue to constitute realizations (or simulations) of abstract membrane computations as illustrated here.* (A suggested alternate biochemical procedure for realizing operations such as e' and d' is methylation as discussed in [H'01b].)

In Section 5, Rules 6 through 11 will be applied. It will be clear that the aqueous realizations of Rules 7, 8, 9 and 10 are almost identical with the realizations of Rules 1, 2, 3 and 4, respectively. The fundamental distinction lies only in the replacement of the singly primed operation symbols in the cycle 1,2,3,4 by the doubly primed operation symbols in the cycle 7,8,9,10. The distinction is simply that, whereas a' deletes *the symbol* a from each string in M, a" deletes from N *each string* in which the symbol a occurs. The distinction between these two operations has lead to the use of the nesting of membrane M inside N. Rule 6 merely symbolizes the transition between the two stages of the computation and Rule 11 indicates the completion of the computation in the same manner in which Rule 6 indicated the conclusion of the computation in Section 3. In [HYG'99] [YHG'00] the double primed operations were implemented using restriction enzymes, bacterial amplification, but *no ligase*.

5. A Membrane System for the Satisfiability Problem.

We have reported in [HYG'99] and [YHG'00] a planned wet lab aqueous solution of the instance of the Satisfiability Problem (SAT) [GJ'79] that we treat here. In the months following the appearance of [HYG'99] we successfully completed this computation in the manner described there and in [YHG'00]. Gel photos confirming this successful solution will be supplied on request.

Let A, B, and C be three Boolean variables. Let a, b, and c denote the negations of A, B, and C, respectively. Consider the set of four clauses: A OR B,   a OR B OR c,   b OR c,   a OR C. For this instance of the satisfiability problem we ask whether there is a truth value setting of the Boolean variables A, B, and C for which the four clauses all evaluate to T (True).

We place the six literals in a membrane M[0], expressed as the string AaBbCc. We then place M[0] and its content into a membrane N[0] along with three pairs expressed as strings Aa, Bb, Cc. (Motive: Within N[0] the three contradictions: A and a, B and b, C and c will be eliminated precisely as though they were three edges in an independence problem as in Section 3). We then place N[0] and its contents into SK along with the negations of the four clauses A OR B, a OR B OR c, b OR c, a OR C expressed as the strings ab, AbC, BC, Ac. (Motive: Within SK the content of N[1] will initially be the eight consistent truth settings for the six literals; each truth setting that fails to provide the value T to a first (respectively second, third, fourth) clause will then be deleted from the membrane N.)   This gives the initial line of the computation below.

The computation proceeds through the applications of the sequence of thirteen rules $(1,2,3,4)^3 6$ followed by the sequence of seventeen rules $(7,8,9,10)^4 11$. The computations terminates and yields the single string aBc which indicates that there is a unique truth setting for the three variables A, B, and C that provides the value T (True) for each of the four clauses. This truth setting is: A = False, B = True, and C = False.

SK{ab, AbC, BC, Ac,   N[0]{Aa, Bb, Cc,   M[0]{AaBbCc}}}
        SK{Clauses, N[0]{Aa, Bb,
                M[1,Cc]{AaBbCc}}}   Where: Clauses = ab, AbC, BC, Ac
        SK{Clauses, N[0]{Aa, Bb,
                M[1,−c]{c'(AaBbCc)}, M[1,C−]{C'(AaBbCc)}}}
        SK{Clauses, N[0]{Aa, Bb,
                M[2,−c]{AaBbC}, M[2,C−]{AaBbc}}}
SK{Clauses, N[0]{Aa, Bb, M[0]{AaBbC, AaBbc}}}
        SK{Clauses, N[0]{Aa,
                M[1,Bb]{AaBbC, AaBbc}}}
        SK{Clauses, N[0]{Aa,
                M[1,−b]{b'(AaBbC, AaBbc)}, M[1,B−]{B'(AaBbC, AaBbc)}}}

SK{Clauses, N[0]{Aa,
             M[2,–b]{AaBC, AaBc}, M[2,B–]{AabC, Aabc}}}
SK{Clauses, N[0]{Aa, M[0]{AaBC, AaBc, AabC, Aabc}}}
        SK{Clauses, N[0]{
             M[1,Aa]{AaBC, AaBc, AabC, Aabc}}}
        SK{Clauses, N[0]{
             M[1,–a]{a'(AaBC, AaBc, AabC, Aabc)}, M[1,A–]{A'(AaBC, AaBc, AabC, Aabc)}}}
        SK{Clauses, N[0]{
             M[2,–a]{ABC, ABc, AbC, Abc}, M[2,A–]{aBC, aBc, abC, abc}}}
SK{Clauses, N[0]{M[0]{ABC, Abc, AbC, Abc, aBC, aBc, abC, abc}}}
SK{Clauses, N[1]{ABC, Abc, AbC, Abc, aBC, aBc, abC, abc}}
= SK{ab, AbC, BC, Ac, N[1]{ABC, Abc, AbC, Abc, aBC, aBc, abC, abc}}
        SK{ab, AbC, BC, N[2,Ac]{ABC, Abc, AbC, Abc, aBC, aBc, abC, abc}}
        SK{ab, AbC, BC,
             N[2,–c]{c"(ABC, Abc, AbC, Abc, aBC, aBc, abC, abc)},
             N[2,A–]{A"(ABC, Abc, AbC, Abc, aBC, aBc, abC, abc)}}
        SK{ab, AbC, BC, N[3,–c]{ABC, AbC, aBC, abC}}, N[3,A–]{aBC, aBc, abC, abc}}
SK{ab, AbC, BC, N[1]{ABC, AbC, aBC, abC, aBc, abc}}
        SK{ab, AbC, N[2,BC]{ABC, AbC, aBC, abC, aBc, abc}}
        SK{ab, AbC,
             N[2,–C]{C"(ABC, AbC, aBC, abC, aBc, abc)},
             N[2,B–]{B"(ABC, AbC, aBC, abC, aBc, abc)}}
        SK{ab, AbC, N[3,–C]{aBc, abc}, N[3,B–]{AbC, abC, abc}}
SK{ab, AbC, N[1]{aBc, abc, AbC, abC}}
        SK{ab, N[2,AbC]{aBc, abc, AbC, abC}}
        SK{ab, N[2,––C]{C"(aBc, abc, AbC, abC)},
             N[2,–b–]{b"(aBc, abc, AbC, abC)},
             N[2,A––]{A"(aBc, abc, AbC, abC)}}
        SK{ab, N[3,––C]{aBc, abc}, N[3,–b–]{aBc}, N[3,A––]{aBc, abC}}
SK{ab, N[1]{aBc, abc, abC}}
        SK{N[2,ab]{aBc, abc, abC}}
        SK{N[2,–b]{b"(aBc, abc, abC)}, N[2,a–]{a"(aBc, abc, abC)}}
        SK{N[3,–b]{aBc}, N[3,a–]{ }}
SK{N[1]{aBc}}
SK{aBc}.

6. A Compression of the Derivations and a Simplified Imagery.

        Once the correlation between Rules 1,2,3,4 and Rules 7,8,9,10 with their respective aqueous implementations is completely clear, *one can safely delete the indented lines from the derivations given in Sections 3 and 5*. The application of intervening rule sequences, i.e. (1234) and (78910), is implied by the progression of the successive non–indented lines of the derivation. In this sense, the indented lines are redundant. Suppose now that the indented lines in the derivations have been deleted. The imagery for the derivation in Section 3 becomes especially clear: The cell M[0] eats and digests in succession de, cd, bc, and ab. Since M[0]finds no further food available, its cell membrane dissolves and dumps its content into SK, providing the solution. The imagery for the derivation in Section 5 is also clear: The cell M[0] eats and digests in succession Cc, Bb, and Aa. Since M[0] finds no further food available, its cell wall dissolves and dumps the content into N[0], resulting in a change of state to N[1]. The cell N[1] then eats and digests, with a somewhat different metabolism from that of M[0], in succession Ac, BC, AbC, and ab. As N[1] finds no further food available, its cell wall dissolves and dumps the content into SK, providing the solution.

References.

[ACD'01] I. Antoniou, C.S. Calude and M.J. Dineen, Eds., *Unconventional Models of Computation UMC'2K*, Springer, London (2001).

[CCD'98] C.S. Calude, J. Casti & M.J. Dineen, Eds., *Unconventional Models of Computation*, Springer, Singapore (1998).

[CP'01] C.S. Calude & Gh. Paun, *Computing with Cells and Atoms – An Introduction to Quantum, DNA and Membrane Computing*, Taylor & Francis, London, (2001).

[Fweb] http://www.liacs.nl/home/pier/webPagesDNA

[GJ'79] M.R.Garey & D.S. Johnson, *Computers and Intractibility – A Guide to the Theory of NP−Completeness*, Freeman, New York (1979).

[H'00] T. Head, Circular suggestions for DNA computing, in: *Pattern Formation in Biology, Vision and Dynamics,* Ed. By A. Carbone, M. Gromov & P. Prusinkiewicz (2000)325−335.

[H'01a] T. Head, Splicing systems, aqueous computing, and beyond, in: *Unconventional Models of Computation UMC'2K*, Ed. by I. Antoniou, C.S. Calude and M.J. Dineen, Springer, London, (2001).

[H'01b] T. Head, Writing by methylation proposed for aqueous computing, Chapter 31 of: *Where Mathematics, Computer Science, Linguistics and Biology Meet*, Ed. by C. Martin−Vide & V. Mitrana, (2001)353−360.

[HYG'99] T. Head, M. Yamamura & S. Gal, Aqueous computing: writing on molecules, in: *Proc. Congress on Evolutionary Computation* 1999, IEEE Service Center, Piscataway, NJ (1999)1006−1010.

[HRBBLS'00] T. Head, G. Rozenberg, R. Bladergroen, C.K.D. Breek, P.H.M. Lomerese & H. Spaink, *Bio Systems* 57(2000)87−93.

[OKLL'97] Q. Ouyang, P.D. Kaplan, P.D.S. Liu & A. Libchaber, DNA solution of the maximal clique problem, *Science* 278(1997)446−449.

[P'98] Gh. Paun, Ed., *Computing with Biomolecules – Theory and Experiments*, Springer−Verlag, Berlin (1998).

[P'00] Gh. Paun, Computing with membranes, *J. Computer & System Sciences*, 61(2000)108−143.

[PRS'98] Gh. Paun, G. Rozenberg & A. Salomaa, *DNA Computing – New Computing Paradigms*, Springer−Verlag, Berlin (1998).

[Zweb] http://bioinformatics.bio.disco.unimib.it/psystems

[YHG'00] M. Yamamura, T. Head & S. Gal, Aqueous computing – mathematical principles of molecular memory and its biomolecular implementation, Chapter 2 in: *Genetic Algorithms 4*, Ed. By H. Kitano, (2000)49−73 (in Japanese).