

Organization-Oriented Chemical Programming

Peter Dittrich and Naoki Matsumaru
Friedrich Schiller University Jena
Department of Mathematics and Computer Science
Bio Systems Analysis Group
07743 Jena, Germany
dittrich/naoki@minet.uni-jena.de

Abstract

Chemical information processing possesses a variety of valuable properties, such as, robustness, concurrency, fault-tolerance, and evolvability. However, it is difficult to predict and program a chemical system, because the computation emerges as a global phenomenon from microscopic reactions. Here, we will present design principles for chemical programs. We focus on programs that should compute a qualitative and not quantitative result. The design principles are based on chemical organization theory, which defines a chemical organization as a closed and self-maintaining set of molecular species. The fundamental assumption of so called organization-oriented programming is that computation should be understood as a movement between chemical organizations. In this case we expect that the resulting system is more robust, and fine-tuning of the kinetic laws will be less important.

1. Introduction

Information processing by chemical reactions can be found in all living systems. It is known to be robust, self-organizing, adaptive, decentralized, asynchronous, fault-tolerant, and evolvable. These valuable properties are exploited by a variety of approaches using real molecules [2, 6] or artificial molecules *in-silico* [4, 16]. When taking real molecules, one aims at exploring new substrates for computation [23]. In *in-silico* chemical computing the chemical metaphor is utilized to program or to build computational systems [8]. In this case the chemical metaphor serves as a design principle for new software or hardware architectures built on conventional silicon devices. Examples are, chemical-like formal systems that model concurrent processes, e.g., Gamma [4], CHAM [5], or P-Systems [18]; or new architectures inspired by chemistry, e.g., reaction-diffusion processors [1].

Chemical programming requires to define microscopic elements that lead to a global behavior realizing a desired computation. Chemical programming can be decomposed into different steps: choosing the molecules, reaction rules, kinetics, and environmental conditions. Note that when using real molecules all these steps are strongly interconnected, e.g., after choosing the molecules the set of reaction rules is also determined; the kinetics might be influenced by temperature, spatial structures, or other environmental conditions. In *in-silico* (or artificial) chemical computing we can define molecules, reaction rules, and kinetics quite independently.

Because the computation emerges from an interplay of many microscopic interactions, it is, in general, difficult to find the right rules for a desired behavior. In general, the relation between microscopic reaction rules and the resulting behavior is non-trivial and non-linear, and therefore usually difficult to predict. However, predictability is a necessary condition for programming manually [22].

Like conventional programming chemical programming should be guided by principles and metaphors, which possess a theoretical base. These theoretical principles may provide recipes, which improve coherence and predictability.

Currently it seems that no universal theory is in sight. Therefore a plethora of theoretical principles should be in our toolbox. One such tool is chemical organization theory [9]. The theory allows to relate reaction rules to the potential behavior they generate. The theory focuses on the qualitative change of a chemical system (i.e., the change of the chemical species present) and abstracts from quantitative change (i.e., a change in concentration). A central concept of the theory is the chemical organization, which is a set of molecular species that is closed and self-maintaining [13].

The fundamental assumption of so called organization-oriented programming is that computation should be understood as a movement between chemical organizations.

When following an organization-oriented approach, we concentrate first on the reaction network neglecting kinetic laws. The reaction network is designed with respect to its organizational structure. Then, in a second step, the kinetics is specified, which determines the dynamics between and inside organizations. The underlying hypothesis is that when a computation can be explained as a movement between organizations, it is more robust and fine-tuning of the kinetics is less important.

Before describing the design principles we give a brief introduction to chemical organization theory. Finally we will demonstrate our approach by discussing chemical boolean logic from an organization-oriented point of view.

2. Chemical Organization Theory

Chemical organization theory deals with reaction networks. A reaction network $\langle \mathcal{M}, \mathcal{R} \rangle$ consists of a set of molecular species \mathcal{M} and a set of reaction rules $\mathcal{R} \subseteq \mathcal{P}_M(\mathcal{M}) \times \mathcal{P}_M(\mathcal{M})$, where $\mathcal{P}_M(\mathcal{M})$ denotes the set of all multisets with elements from \mathcal{M} . A multiset differs from an ordinary set in that it can contain multiple copies of the same element. A reaction rule is similar to a rewriting operation [21, 3] on a multiset. Adopting the notion from chemistry, a reaction rule $(A, B) \in \mathcal{R}$ is written as $A \rightarrow B$ where both A and B are multisets of molecular species. The elements of each multiset are listed with “+” symbols between them. Instead of writing $\{s_1, s_2, \dots, s_n\}$, the set is written as $s_1 + s_2 + \dots + s_n$ in the context of reaction rules. We also rewrite $a + a \rightarrow b$ to $2a \rightarrow b$ for simplicity. Note that “+” is not an operator but a separator of elements.

A set of molecular species is called an organization if the following two properties are satisfied: closure and self-maintenance. A set of molecular species is closed when all reaction rules applicable to the set cannot produce a molecular species that is not in the set. This is similar to the algebraic closure of an operator in set theory.

Definition 1 (closure [12]). *Given an reaction network $\langle \mathcal{M}, \mathcal{R} \rangle$, a set of molecular species $C \subseteq \mathcal{M}$ is closed, if for every reaction $(A \rightarrow B) \in \mathcal{R}$ with $A \in \mathcal{P}_M(C)$, also $B \in \mathcal{P}_M(C)$ holds.*

The second important property, self-maintenance, assures, roughly speaking, that all molecules that are consumed within a self-maintaining set can also be produced by some reaction pathways within the self-maintaining set. The general definition of self-maintenance is more complicated than the definition of closure because the production and consumption of a molecular species can depend on many molecular species operating as a whole in a complex pathway.

Definition 2 (self-maintenance [9]). *Given an reaction network $\langle \mathcal{M}, \mathcal{R} \rangle$, let i denote the i -th molecular species of \mathcal{M}*

and the j -th reaction rules is $(A_j \rightarrow B_j) \in \mathcal{R}$. Given the stoichiometric matrix $\mathbf{M} = (m_{i,j})$ that corresponds to $\langle \mathcal{M}, \mathcal{R} \rangle$ where $m_{i,j}$ denotes the number of molecules of species i produced¹ in reaction j , a set of molecular species $S \subseteq \mathcal{M}$ is self-maintaining, if there exists a flux vector $\mathbf{v} = (v_{A_1 \rightarrow B_1}, \dots, v_{A_j \rightarrow B_j}, \dots, v_{A_{|\mathcal{R}|} \rightarrow B_{|\mathcal{R}|}})^T$ satisfying the following three conditions:

1. $v_{A_j \rightarrow B_j} > 0$ if $A_j \in \mathcal{P}_M(S)$,
2. $v_{A_j \rightarrow B_j} = 0$ if $A_j \notin \mathcal{P}_M(S)$,
3. $(\mathbf{M}\mathbf{v})_i \geq 0$ if $s_i \in S$.

These three conditions can be read as follows: When the j -th reaction is applicable to the set S , the flux $v_{A_j \rightarrow B_j}$ must be positive (Condition 1). All other fluxes are set to zero (Condition 2). Finally, the production rate $(\mathbf{M}\mathbf{v})_i$ for all the molecular species $s_i \in S$ must be nonnegative (Condition 3). Note that we have to find only one such flux vector in order to show that a set is self-maintaining.

Taking closure and self-maintenance together, we arrive at an organization:

Definition 3 (organization [9, 12]). *A set of molecular species $O \subseteq \mathcal{M}$ that is closed and self-maintaining is called an organization.*

We visualize the set of all organizations by a Hasse diagram, in which organizations are arranged vertically according to their size in terms of the number of their members (e.g., Fig. 1). Two organizations are connected by a line if the lower organization is contained in the organization above and there is no other organization in between.

Finally, a relevant theorem from Ref. [9] states that given a differential equation describing the dynamics of a chemical reaction system and the reaction network corresponding to that system, then the set of molecular species with positive concentrations in a fixed point (*i.e.*, stationary state), if it exists, is an organization. In other words, we can only obtain a stationary behavior with a set of molecular species that are both closed and self-maintaining.

3. Organization-Oriented Design Principles

3.1. Computing Within vs. In-Between Organizations

Chemical computing can be distinguished whether a computation takes place within one organization or whether the computation can be explained as a movement between

¹Formally, this can be defined as $m_{i,j} = \#(i \in B_j) - \#(i \in A_j)$, where $\#(i \in A_j)$ denotes the number of occurrences of species i on the left-hand side of reaction j and $\#(i \in B_j)$ the number of occurrences of species i on the right-hand side of reaction j .

organizations. Computation within one organization might exploit bistability of a chemical system or may implement a continuous function, e.g., a chemical neuron [14] or a chemical square-root [7]. Computation as a movement between organizations is characterized by the fact that the molecular species present in the reaction vessel change over time. This is, for example, the case in classical DNA Computing [2], which can be understood in terms of chemical organization theory: For each solution there is at least one organization and the experimental steps assure that the system will end up in such an organization.

3.2. Design Principles

Let us proceed now with preliminary design principles inspired by chemical organization theory:

P1: There should be (at least) one organization for each behavioral output class. Following organization-oriented design principles makes sense when computation appears as a movement in-between organizations. In that case the output behavior can be categorized in different discrete behavioral classes. Here we demand that different outputs should be represented by different organizations. For example, in a chemical flip-flop [17], these behavioral classes are the different states of the flip-flop, and we demand that each state should be represented by a different organization.

P2: The set of molecular species (and the organization) representing a result should be in the closure of the species representing the initial input. This principle assures that there is a reaction path leading from the input to the desired output species. The desired output set must be (contained in) a self-maintaining set within the closure of the initial input configuration. Ideally, within the closure there is a largest self-maintaining set representing the output. Otherwise the dynamics may stuck above the desired output set.

P3: The set of molecules (and certain environmental conditions) representing an input should generate the organization representing the desired output.

Given a set of (input) species A we can generate an organization O by first adding all possible reaction products until we reach a closed set C . Then we remove species until we reach a largest self-maintaining set S contained in C . In a specific class of networks (called semi-consistent [10]), S is unique. And in so called consistent networks [10], S is additional closed, thus an organization. In chemical computing it is not necessary that the generate operation is unique. Moreover it can even be beneficial, if it is not unique. As for example in the chemical FLIP-FLOP [17].

Note that even if the input configuration generates the desired output organization (i.e., a set of chemical species), this does not guarantee that we will end up in that desired

organization. There might be kinetic laws such that we will end up in an organization below the desired organization. Further note that this fact might be used by a computation, in which case P3 would be violated.

P4: Eliminate organizations not representing a desired output. If each organization represents a desired output, the system's dynamics must converge to a set of chemical species representing an output. Therefore it makes obviously sense to eliminate an organization not representing an output. This can be achieved by either destroying its closure property or its self-maintenance. Note, however, that in general not all such organizations can be eliminated (see, e.g., the chemical FLIP-FLOP [17]).

P5: An output organization should have no organization below.

As mentioned above, if there is an organization below an output organizations, the system might move-down spontaneously, leaving the output organization.

P5: Assure, if possible, stoichiometrically the stability of an output organization.

If an output organization contains another organization, the system state can move spontaneously down to this organization. In some cases this down-movement can be ruled out by a purely stoichiometric argument. That is, we can design the reaction network such that for any kinetic law the organization is stable. As a simple example consider the system $\mathcal{R} = \{a \rightarrow b, b \rightarrow a\}$, which has two organizations. Due to mass-conservation, the system can never move spontaneously from organization $\{a, b\}$ to the empty organization.

P6: Use kinetic laws for fine tuning. The kinetic laws must assure that the output organizations are stable. Furthermore the kinetics determines the transition dynamics between organizations. Finding the right kinetic laws is a non-trivial task, because an output organization usually contains other organizations (ie., there are organizations below). Chemical organization theory assures that such laws exists (to a certain extend). For finding them we have to rely on classical dynamical systems theory. However, note that it is possible to derive at least in some cases rigorously dynamical stability from network structure [11]. Further note that there can be a tradeoff between stability of an organization and the speed of computation. Finally, the influence of the kinetic laws can be studied by mapping the quantitative dynamics (i.e., trajectory in the concentration space) to the Hasse-diagram of organizations (like in Ref. [10], Figure 4).

4. Organization-Oriented Chemical Boolean Logics

In this section we will demonstrate the approach by studying boolean logics implemented by chemical reaction systems. First, we investigate two different codings from an

organization point of view. Then, a recipe for programming boolean logics is described.

4.1. Functions like NOT and XOR Require Non-trivial Encoding

The first code (**Code 1**, Table 1) assumes that a logic one (true) is trivially encoded by the presence of a molecular species, while logic zero (false) is encoded by the same species' absence. We will show that in this case there are no reaction rules that implement an organization-oriented XOR, i.e., a chemical XOR-network whose behavior can be explained purely from an organization-oriented point of view. When we assume alternatively a code where each logical variable is represented by two molecules (**Code 2**, next section), we can define such an XOR gate.

In order to prove that Code 1 is not sufficient for an organization-oriented XOR, it is sufficient to show that we can even not construct an organization-oriented NOT with Code 1: Let $x \in \mathcal{M}$ and $y \in \mathcal{M}$ be the species representing the input and output of the NOT-gate, respectively. For x being not present, the organizations of the network must contain y and not x . For x being present (i.e., $(\rightarrow x) \in \mathcal{R}$), there should be one or more organizations, all of which must not contain y . We distinguish now two cases: Case 1: y cannot be overproduced (i.e., a decay of y cannot be compensated. by the network). When x appears (input 1), x must destroy y in order to obtain output 0. However, when x disappears (input 0), y can not be regenerated, because we assumed that it cannot be overproduced. Case 2: y can be overproduced. When x appears (input 1), y cannot vanish completely, because a decay initiated by x can be compensated by the remaining network. In case the production y requires a species that is not overproducible, we obtain Case 1. q.e.d.

Therefore, with Code 1 we can only build a NOT-gate that functions once. That is, we would have to regenerate y by an external "clock" after each operation. In the following recipe we will use a different code. Code 2 requires two molecular species for each logical variable, which allows to build arbitrary boolean functions. Note that in biological signaling networks signals are often encoded by two (or more) molecules, for example, by a phosphorylated and unphosphorylated protein. Whether this is related to our organization-oriented design remains speculative.

4.2. Recipe for Organization-Oriented Programming of Chemical Logics

In this section we present a procedure for designing chemical reaction networks implementing a logic circuit (see Table 2 for a recipe and Ref. [17] for details). A logic circuit is a composition of logic gates. As such it can be

Code 1		Code 2	
value	representation	value	representation
$b = 0$	$[b] = 0$	$b = 0$	$[b] > 0, [B] = 0$
$b = 1$	$[b] > 0$	$b = 1$	$[b] = 0, [B] > 0$

Table 1. Two codes for a boolean variable b using one molecular species $\{b\}$ and two molecular species $\{b, B\}$, respectively.

Input: Boolean network given by two sets: a set of M boolean functions $\{F_1, \dots, F_M\}$ and a set of N boolean variables $\{b_1, \dots, b_M, \dots, b_N\}$. Variables $\{b_1, \dots, b_M\}$ are determined by the boolean functions (*internal variables*); the remaining variables $\{b_{M+1}, \dots, b_N\}$ are input variables of the boolean network.

Output: Reaction network $\langle \mathcal{M}, \mathcal{R} \rangle$ (a set of molecular species \mathcal{M} and a set of reaction rules \mathcal{R}) representing the boolean network without any input variable specified.^a

Algorithm:

- For each boolean variable b_j :
 - Add two molecular species, b_j and B_j , to \mathcal{M} ;^b
 - Add one *destructive reaction* of the form $b_j + B_j \rightarrow \emptyset$ to \mathcal{R} ;
- For each boolean function F_i :
 - Create the truth table of F_i with 2^{n_i} input cases (where n_i is the arity of F_i);
 - For each input case, create a *logical reaction*.^c
 - Lefthand side (*reactants*) corresponds to the input of F_i .
 - Righthand side (*products*) consists of one molecular species representing the respective boolean output of F_i .

^aSpecifying an input variable of the boolean network is coded by an inflow reaction.

^bAs a naming convention of molecular species in this paper, the lowercase species represents value 0 in the boolean variable, and the uppercase stands for 1.

^cFor example, the XOR-function is converted into reactions as follows:

b_2	b_3	$b_1 = F_1(b_2, b_3)$		Reactants	\rightarrow	Products
0	0	0	\Rightarrow	$b_2 + b_3$	\rightarrow	b_1
0	1	1		$b_2 + B_3$	\rightarrow	B_1
1	0	1		$B_2 + b_3$	\rightarrow	B_1
1	1	0		$B_2 + B_3$	\rightarrow	b_1

Table 2. Recipe for mapping a boolean circuit to a chemical reaction network.

fully described by a set of boolean functions and boolean variables, forming a boolean network [15]. Let the boolean network be defined by a set of M boolean functions and a set of N ($\geq M$) boolean variables:

$$\{b_1, \dots, b_M, \dots, b_N\} \quad (1)$$

where $\{b_j | 1 \leq j \leq M\}$ are determined by the boolean functions (*internal variables*) and the remaining variables $\{b_j | M < j \leq N\}$ are the input variables of the boolean network. The set of boolean functions is

$$\{b_i = F_i(b_{q(i,1)}, \dots, b_{q(i,n_i)}) \mid i = 1, \dots, M\} \quad (2)$$

where $b_{q(i,k)}$ indicates the boolean variable listed as the k -th argument of the i -th function. Since the i -th boolean function F_i takes n_i boolean variables as arguments, there are 2^{n_i} possible inputs. Thus the truth table T_i for function F_i has 2^{n_i} rows and $n_i + 1$ columns:

$$T_i : \begin{bmatrix} t_{1,1}^i & \cdots & t_{1,n_i}^i & t_{1,n_i+1}^i \\ \vdots & \ddots & \vdots & \vdots \\ t_{2^{n_i},1}^i & \cdots & t_{2^{n_i},n_i}^i & t_{2^{n_i},n_i+1}^i \end{bmatrix} \quad (3)$$

where $t_{h,k}^i \in \{0, 1\}$ is the boolean value of the k -th argument in the h -th input case for the i -th boolean function. The $(n_i + 1)$ -th column contains the output of F_i .

Given the boolean network, a reaction network $\langle \mathcal{M}, \mathcal{R} \rangle$ is designed as described by Table 2. The resulting reaction network $\langle \mathcal{M}, \mathcal{R} \rangle$ implements the logic circuit without any input specified. The input variables of the boolean network $\{b_j | M < j \leq N\}$ must be initialized externally because they are not set by the boolean functions. The initialization of the input variables is encoded by an inflow reaction, which is a zero-order reaction producing substances from the empty set. If an input variable b_j is initialized to 0, for example, the reaction network is changed to $\langle \mathcal{M}, (\mathcal{R} \cup \{\emptyset \rightarrow s_{2j-1}\}) \rangle$. It is possible for more than one variable to be initialized in this manner as it is possible for more than one molecular species to be injected by the influx.

4.3. Example: Chemical XOR

For an XOR logic gate we need two boolean input variables a, b and one boolean output variable c . The logic function $c = F_c(a, b)$ is given by the truth table

$$T_c : \begin{array}{c} \begin{array}{ccc} a & b & c \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{array} \end{array} \quad (4)$$

Given the definition of the XOR boolean network, a reaction network $\langle \mathcal{M}_{\text{XOR}}, \mathcal{R}_{\text{XOR}} \rangle$ is generated to implement the

logic gate. Since there are $N = 3$ boolean variables, the set of molecular species consists of six molecular species:

$$\mathcal{M}_{\text{XOR}} = \{a, A, b, B, c, C\} \quad (5)$$

where the lower- and uppercase version of the variable name are assigned to the boolean variable of that name. For example, molecular species a represents boolean variable $a = 0$, and A stands for $a = 1$ (see Code 2, Table 1).

The set of reaction rules \mathcal{R}_{XOR} is defined following the recipe of Table 2 as:

$$\mathcal{R}_{\text{XOR}} = \{a + b \rightarrow c, a + B \rightarrow C, A + b \rightarrow C, A + B \rightarrow c, a + A \rightarrow \emptyset, b + B \rightarrow \emptyset, c + C \rightarrow \emptyset\}.$$

The reaction network $\langle \mathcal{M}_{\text{XOR}}, \mathcal{R}_{\text{XOR}} \rangle$ implements the XOR logic gate without any input specified. There are 15 organizations [17]. Now we set only one input variable of the boolean network by adding an inflow reaction to the set of reaction rules (Figure 1). As we can see, the set of organizations reduces to three. There is always one organization containing c and one containing C , which means that with one input the output is not determined from an organization-oriented point of view.

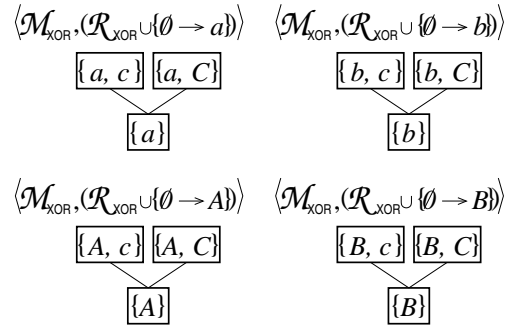


Figure 1. Four Hasse-diagrams of the chemical xor for underspecified inputs. Only one input signal is specified by an inflow reaction.

When we finally provide both inputs, the Hasse diagram of organizations collapses so that only one organization remains for every input condition (Figure 2). This implies that, no matter how we chose the kinetic details, no other molecular species than those of the organization can be sustained in the reaction vessel regardless of the initial state. We can also see that the remaining organization contains the desired output molecular species c or C , respectively.

5. Conclusion and Outlook

Organization-oriented programming guides the design of the reaction network of a chemical program. When the

$$\langle \mathcal{M}_{\text{XOR}}, (\mathcal{R}_{\text{XOR}} \cup \{\emptyset \rightarrow a, \emptyset \rightarrow b\}) \rangle: \boxed{\{a, b, c\}}$$

$$\langle \mathcal{M}_{\text{XOR}}, (\mathcal{R}_{\text{XOR}} \cup \{\emptyset \rightarrow a, \emptyset \rightarrow B\}) \rangle: \boxed{\{a, B, C\}}$$

$$\langle \mathcal{M}_{\text{XOR}}, (\mathcal{R}_{\text{XOR}} \cup \{\emptyset \rightarrow A, \emptyset \rightarrow b\}) \rangle: \boxed{\{A, b, C\}}$$

$$\langle \mathcal{M}_{\text{XOR}}, (\mathcal{R}_{\text{XOR}} \cup \{\emptyset \rightarrow A, \emptyset \rightarrow B\}) \rangle: \boxed{\{A, B, c\}}$$

Figure 2. Four Hasse-diagrams of the chemical xor for each possible input. Both inputs are specified by inflow reactions.

chemical computation can be explained as movement between organizations it is more robust, because kinetics is less important. In certain situations (e.g. chemical XOR) the theory can show that the system is completely robust such that the computation is always achieved independently of the kinetics.

The organizational structure does not allow to derive kinetic properties like stability. For this we have to rely on other theories (e.g., Refs. [11, 19, 20]) Integrating those with an organizational analysis would be interesting.

The basic definitions of the theory are also applicable to implicitly defined reaction systems, where molecules possess a structure and where organization may contain an infinite amount of molecules. Implicit representations of organizations is especially important when dealing with combinatorial explosions like those encountered in polymer systems. However developing concepts, tools, and software to represent and handle organizations implicitly has yet to be done, which is an important task for future research.

Acknowledgment

We thank Bashar Ibrahim for valuable comments. We acknowledge financial support by the German Research Foundation (DFG) Grant DI 852/4.

References

- [1] A. Adamatzky. Universal dynamical computation in multidimensional excitable lattices. *Int. J. Theor. Phys.*, 37(12):3069–3108, 1998.
- [2] L. M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021, 1994.
- [3] J.-P. Banâtre, P. Fradet, and Y. Radenac. Principles of chemical programming. In S. Abdennadher and C. Ringeissen, editors, *RULE'04 Fifth International Workshop on Rule-Based Programming*, pages 98–108. Tech. Rep. AIB-2004-04, Dept. of Comp. Sci., RWTH Aachen, Germany, 2004.
- [4] J.-P. Banâtre and D. L. Métayer. The GAMMA model and its discipline of programming. *Sci. Comput. Program.*, 15(1):55–77, 1990.
- [5] G. Berry and G. Boudol. The chemical abstract machine. *Theor. Comput. Sci.*, 96(1):217–248, 1992.
- [6] M. Conrad. Information processing in molecular systems. *Currents in Modern Biology*, 5:1–14, 1972.
- [7] S. H. Deckard A. Preliminary studies on the in silico evolution of biochemical networks. *Chembiochem.*, 5(10):1423–1431, 2004.
- [8] P. Dittrich. Chemical computing. In J.-P. Banâtre, J.-L. Giavitto, P. Fradet, and O. Michel, editors, *Unconventional Programming Paradigms (UPP 2004)*, volume 3566 of *LNCIS*, pages 19–32. Springer, Berlin, 2005.
- [9] P. Dittrich and P. Speroni di Fenizio. Chemical organization theory. *Bull. Math. Biol.*, 69(4):1199–1231, 2007.
- [10] P. Dittrich and P. Speroni di Fenizio. Chemical organization theory. *Bull. Math. Biol.*, 69(4):1199–1231, 2007.
- [11] M. Feinberg and F. J. M. Horn. Dynamics of open chemical systems and the algebraic structure of the underlying reaction network. *Chem. Eng. Sci.*, 29:775–787, 1974.
- [12] W. Fontana and L. W. Buss. 'The arrival of the fittest': Toward a theory of biological organization. *Bull. Math. Biol.*, 56:1–64, 1994.
- [13] W. Fontana and L. W. Buss. 'The arrival of the fittest': Toward a theory of biological organization. *Bull. Math. Biol.*, 56:1–64, 1994.
- [14] A. Hjelmfelt, E. D. Weinberger, and J. Ross. Chemical implementation of neural networks and turing machines. *Proc. Natl. Acad. Sci. USA*, 88:10983–10987, 1991.
- [15] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.*, 22:437–467, 1969.
- [16] R. Laing. Some alternative reproductive strategies in artificial molecular machines. *J. Theor. Biol.*, 54:63–84, 1975.
- [17] N. Matsumaru, F. Centler, P. S. di Fenizio, and P. Dittrich. Chemical organization theory as a theoretical base for chemical computing. In *International Journal on Unconventional Computing*, 2007.
- [18] G. Păun. Computing with membranes. *J. Comput. Syst. Sci.*, 61(1):108–143, 2000.
- [19] A. Sensse and M. Eiswirth. Feedback loops for chaos in activator-inhibitor systems. *J. Chem. Phys.*, 122(4):44516, 2005.
- [20] R. Steuer, A. N. Nesi, A. R. Fernie, T. Gross, B. Blasius, and J. Selbig. From structure to dynamics of metabolic pathways: application to the plant mitochondrial TCA cycle. *Bioinformatics*, 23(11):1378–1385, 2007.
- [21] Y. Suzuki and H. Tanaka. Symbolic chemical system based on abstract rewriting system and its behavior pattern. *Artificial Life and Robotics*, 1(4):211–219, 1997.
- [22] K.-P. Zauner. From prescriptive programming of solid-state devices to orchestrated self-organisation of informed matter. In J.-P. Banâtre, J.-L. Giavitto, P. Fradet, and O. Michel, editors, *Unconventional Programming Paradigms: International Workshop UPP 2004*, volume 3566 of *LNCIS*, pages 47–55. Springer, Berlin, 2005.
- [23] K.-P. Zauner. Molecular information technology. *Cr. Rev. Sol. State*, 30(1):33–69, 2005.