in: Unconventional Programming Paradigms (UPP 2004),
J.-P. Banatre, J.-L. Giavitto, P. Fradet, O. Michel, Editors,
LNCS 3566, pp. 19-32, Springer, Berlin, 2005

Chemical Computing

Peter Dittrich

Bio Systems Analysis Group
Jena Centre for Bioinformatics (JCB) and
Department of Mathematics and Computer Science
Friedrich-Schiller-University Jena
D-07743 Jena, Germany
http://www.minet.uni-jena.de/csb/

Abstract. All information processing systems found in living organisms are based on chemical processes. Harnessing the power of chemistry for computing might lead to a new unifying paradigm coping with the rapidly increasing complexity and autonomy of computational systems. Chemical computing refers to computing with real molecules as well as to programming electronic devices using principles taken from chemistry. The paper focuses on the latter, called artificial chemical computing, and discusses several aspects of how the metaphor of chemistry can be employed to build technical information processing systems. In these systems, computation emerges out of an interplay of many decentralized relatively simple components analogized to molecules. Chemical programming encompassed then the definition of molecules, reaction rules, and the topology and dynamics of the reaction space. Due to the self-organizing nature of chemical dynamics, new programming methods are required. Potential approaches for chemical programming are discussed and a road map for developing chemical computing into a unifying and well grounded approach is sketched.

1 Introduction

All known life forms process information on a molecular level. Examples are: signal processing in bacteria (e.g., chemotaxis), gene expression and morphogenesis, defense coordination and adaptation in the immune system, and information broadcasting by the endocrine system. Chemical processes play also an important role, when an ant colony seeks a suitable route to a food source. This kind of chemical information processing is known to be robust, self-organizing, adaptive, decentralized, asynchronous, fault-tolerant, and evolvable. Computation emerges out of an orchestrated interplay of many decentralized relatively simple components (molecules).

How can chemistry be employed for computing? First, it should be noted that chemistry is used for the fabrication of electronic devices. However, here we are interested in approaches where chemistry stimulates the development of new computational paradigms. These approaches can be distinguished according to the following two dimensions: First, *real chemical computing* where real molecules and real chemical processes are employed to compute. Second, *artificial chemical computing* where the chemical metaphor is utilized to program or to build computational systems. The former aims at harnessing new substrates for computation. The latter takes the chemical metaphor as a design principle for new software or hardware architectures built on conventional silicon devices. So, artificial chemical computing includes constructing chemical-like formal system in order to model and master concurrent processes, e.g., Gamma [1], CHAM [2]; using the chemical metaphor as a new way to program conventional computers including distributed systems, e.g., smart dust; and taking the chemical metaphor as an inspiration for new architectures, e.g., reaction-diffusion processors [3].

1.1 The Chemical Metaphor

Chemistry is a science of experiment and observation, which provides a particular view on our world. Like physics, chemistry deals with matter and energy, but focuses on substances composed of molecules and how the composition of these substances is changed by "chemical" reactions. Compared with chemistry, physics is more concerned with energy, forces, and motion, ie. the physical change of a system.

Chemistry looks at the macro and micro level: On the macro level emergent properties and the emergent behavior of substances are studied, e.g., color or smell. On the microscopic level, molecular structures and reaction mechanisms are postulated, which are taken to explain macroscopic observations. Ideally, microscopic models allow to formally derive macroscopic observations. However, this is possible only in limited cases, e.g., no algorithm that computes the melting temperature of a molecule given its structure is known. In general, chemistry explains a chemical observation using a mixture of microscopic and macroscopic explanations.

The difficulty to predict the macroscopic behavior from microscopic details has its root in the nature of emergence. The time-evolution of a chemical system is a highly parallel self-organization process where many relatively simple components interact in a nonlinear fashion. And it is a central aim of chemical computing to harness the power inherent in these self-organization processes.

From a computer science perspective it would be quite appealing to achieve computation as an emergent process, where only microscopic rules have to be specified and information processing appears as global behavior. From knowing the biological archetype, we can expect a series of interesting properties, such as, fine grained parallelism without central control, fault tolerance, and evolvability. There is a wide application range, especially where the characteristics of chemical processes fit naturally to the desired task, as for example in highly distributed and dynamic "organic" processor networks or within one computing node to implement particular systems like artificial emotional [4], endocrine [5], or immune systems more naturally. It should be mentioned that chemical processes themselves can be seen as a natural media for information processing either in vitro or in vivo [6–8]; for a recent discussion of molecular computing see ref. [9]. Here we concentrate on how technical electronic systems can utilize the chemical metaphor.

1.2 The Organization of Chemical Explanations

When we study chemistry [10], first we learn how substances look like. We describe macroscopic properties of the substances, such as color, and how substances are composed from elementary objects, the atoms. Second, we learn how substances interact, in particular, we describe the outcome that results from their union. Reactive interactions among molecules require that these molecules come into contact, which can be the result of a collision. Third, we learn the detailed dynamical process of a chemical transformation of substances. All these steps of description can be done on a microscopic and macroscopic level. The steps are also not independent: The properties of substances are often described in terms of how a substance reacts with other substances, e.g., when we say "fluorine is not healthy in large quantities" we describe the property of fluorine by how it interacts with molecules in an organism. In fact, in times when nothing was known about the molecule's structure substances where classified according to their macroscopic appearance and reactive behavior.

Today, classification of substance usually refers to the structure of the molecules, e.g., alcohols are characterized by a functional OH-group. Sometimes only the composition of atoms is taken for classification, e.g., hydrocarbons. Interestingly and importantly for the success of the discipline Chemistry is the fact that structural classification coincide with classifications based on behavior and appearance. This phenomenon is not sufficiently explained by the fact that the function (ie. physical and reactive properties) of a molecule depends on its structure, which is a form of causality. Moreover, similarity in structure tends to coincide with similarity in function, which is a form of *strong* causality between structure and function.

Another important observation should be noted: When we combine some substance in a reaction vessel and wait while these substances react; as a result only a small subset of molecular species will appear, which is usually much smaller than the set of molecular species that could be build from the atoms present in the reaction vessel. So there is also a certain (strong) causality in the dynamics and a dependency on initial conditions. Not everything that is possible does appear, though there is also nondeterminism. So, we can say that a chemical system evolves over time in a contingent way that depends on its history.

1.3 Information Processing and Computing in Natural System

When we intend to take inspiration from chemistry, we have first to investigate where chemical information processing appears in natural systems. Obviously, living systems are prime candidates, since information processing is identified as a fundamental property of life [11].

Information processing in living systems can be observed on at least two different levels: the chemical and the neural level. Where the neural level is responsible for cognitive tasks and fast coherent control, such as vision, planing, and muscle control; chemical information processing is used for regulating and controlling fundamental processes like growth, ontogeny, gene expression, and immune system response. Neurons themselves are based on (electro-)chemical processes, and more often than not, chemical processes are combined with neuronal processes resulting in a large-scale computational result.

Real chemical computing utilizes a series of "chemical principles", which are also relevant for artificial chemical computing, such as: pattern recognition[12], change of conformation[13], chemical kinetics [14], formation of (spatial) structures, energy minimization, and optical activity [15]. Pattern recognition is a central mechanism for explaining reactions among complex biomolecules (e.g., transcription factors binding to DNA). It is also used in real as well as artificial chemical computing approaches, such as DNA computing [12] and rewriting systems [1, 16, 17], respectively.

1.4 Application of the Chemical Metaphor in Computing

There are already a series of approaches in computer science that have been inspired by chemistry: An early example are the *artificial molecular machines* suggested by Laing [18]. These machines consists of molecules (strings of symbols). Each molecule can appear in two forms: data or machine. During a reaction, two molecules come into contact at a particular position. One of the molecules is considered as the active machine, which is able to manipulate the passive data molecule. The primary motivation for developing these molecular machines was to construct artificial organisms in order to develop a general theory for living systems (cf. [19] for a comparing discussion of more recent approaches in that direction).

A fundamentally different motivation has been the starting point for the development of Gamma by Banâtre and Le Métayer, namely to introduce a new programming formalism that allows to automatize reasoning about programs, such as automatic semantic analysis [20, 1]. Gamma is defined by rewriting operations on multisets, which mimics chemical reactions in a well-stirred reaction vessel. Gamma inspired a series of other chemical rewriting systems: Berry and Boudol [2] introduced the *chemical abstract machine* (CHAM) as a tool to model concurrent processes. Păun's P-Systems [16] stress the importance of membranes. Suzuki and Tanaka [21] introduced a rewriting system on multisets in order to study chemical systems, e.g., to investigate the properties of chemical cycles [22], and to model chemical-like systems including economic processes.

Within biological organisms, the endocrine system is a control system, which transmits information by chemical messengers called hormones via a broadcast strategy. The humanoid robot torso COG [5] is an example where the endocrine system has inspired engineering. Artificial hormones are used to achieve a coherent behavior among COG's large number of independent processing elements [5]. In general, chemical-like systems can control the behavior and particularly emotions in artificial agents, e.g., the computer game Creatures [23] and the psychological model PSI by Dörner [4]. Further application areas of chemical computing are: the control of morpho-genetic systems, i.e. the control of morphogenesis by artificial gene expression; in particular, the control of growth of an artificial neural networks (cf. Astor and Adami [24]); and the control of amorphous computers [25]. Finally, Husbands et al. [26] introduced diffusing chemical substances in artificial neural networks (cf. GasNet).

2 Facets of Chemical Computing

As exemplified by the previous section, the world of chemical computing enjoys already a wide spectrum of approaches. This section discusses a set of important aspects, which allow to characterize chemical computing in more detail.

2.1 Microscopic vs. Macroscopic Computing

Chemical information processing can be characterized according to the level on which it appears: In approaches like chemical boolean circuits [27], the chemical neuron [14], or the hypercyclic memory (Sec. 5.2), information is represented by the concentration of substances and computation is carried out by an increase and decrease of concentration levels, which can be regarded as a form of *macroscopic chemical computing.* Alternatively, in *microscopic chemical computing,* the intermediately stored information and computational results are represented by single molecules. Examples are DNA computing [12] and the prime number chemistry (Sec. 5.1). The dynamics is usually stochastic, in contrast to macroscopic computation, which can be more readily described with ordinary differential equations. Nevertheless, microscopic computing also can deliver results virtually deterministically, as shown by the prime number chemistry example in Sec. 5.1.

2.2 Deterministic vs. Stochastic Processes

On the molecular level, chemical processes are stochastic in nature. However, in technical applications deterministic behavior is often required. There are various ways how this can be achieved:

(1) The problem can be stated such that the order of the sequence of collisions does not play a role[1]. An example is the prime number chemistry where we start with a population that contains all numbers between 2 and n. The outcome will be a reactor containing all and only prime numbers less or equal n, independently of the sequence of updates.

(2) Increasing the reactor size would reduce the effect of randomness. If the reactor size and together with it the number of molecules of each molecular type

[1] For a theory that considers the effect of the order of update see "sequential dynamical systems" [28, 29]

tends to infinity, the molecules' concentrations tend to a deterministic dynamics. In this case, the dynamics of the concentrations can be represented by a differential equation and simulated by numerical integration of this equation.

(3) A well-defined deterministic update scheme can be used. For example we can check one reaction rule after another in a fixed predefined sequence, e.g., early ARMS [21] and MGS [17][2] Doing this, we gain determinism and might gain efficiency, but we loose aspects of the chemical metaphor and may introduce artifacts by the update scheme, e.g., when the rule order plays a significant role. This might be reasonable from a computing point of view, but is unnatural from a chemical point of view.

2.3 Closed vs. Open Systems

In thermodynamics, a system that can exchange mass and energy with its environment is called open.When mass is not exchanged the system is called closed If the system cannot exchange anything, it is called isolated. In chemical computing we also encounter closed and open systems, whose characteristics are quite different. In a *closed system*, molecules do not leave the reaction vessel. There is no dilution flow. Reaction rules must be balanced, which means that the mass on the left hand side must be equal to the mass on the right hand side. So, a molecule can only disappear by transforming it via a reaction into other molecules. In an isolated system, stable dissipative structures can not appear; they can only appear as transient phenomena locally. The prime number chemistry is an example for a closed and isolated system. There is no dilution flow and molecules are transformed by the mass-conserving rule: $a + b \rightarrow a + b/a$ for b being a multiple of a.

The hypercyclic memory is an example for an open system. Molecules constantly vanish and are regenerated from an implicitly assumed substrate, which is available at a constant concentration from the environment. Before the query, the system is in a quasi-stationary state, which is a dissipative structure that requires a constant regeneration of all of its components.

The hypercyclic memory is also an example where there is a so called *non-selective dilution flow*, where the rate of decay is proportional to the concentration of a molecule, or more precisely, the concentration of molecules in the dilution flow is the same as in the reaction vessel. Systems with selective dilution flows are not discussed here, but it should be noted that by introducing a selective dilution flow, we can move gradually from an open to a closed system and can capture aspects from both.

Does it make sense to consider open systems with a non-selective dilution flow, where we have to regenerate constantly molecules we wish to have in the reactor? From a formal point of view, both might be equivalent: In an open system, a stable solution is a self-regenerating set of molecules; while in a closed system, a stable solution is just a set of molecules, which do not react further

[2] Note that both mentioned systems (ARMS, MGS) allow also a randomized "natural" update scheme.

to form other molecules (nevertheless there might be a reversible dynamics). So from this point of view, taking a closed systems approach appears more reasonable, because the solution is more stable. We do not have to fear that information gets lost by the dilution flow and we do not have to care for regenerating molecules.

However, when using an open system approach we arrive at more robust and flexible organic systems. Open reaction systems are especially suitable, where the substrate is unreliable and highly dynamic. Consider for example a computational substrate that is under constant change, where nodes are added and removed at a relatively high rate, e.g., the network of activated cellular phones. In such a system, there is no place that exists for long. When a cellular phone is switched off, the molecules residing in that places vanish, too, which causes from a chemical point of view a general, non-selective dilution flow. Thus stable structures must consists of molecules that constantly reproduce themselves as a whole; according to the theory of chemical organization [30], they must encompass a self-maintaining set of molecules.

2.4 Computing with Invisible Networks

What is the difference between chemical computing and an artificial neural network (ANN)? In both approaches, a network is specified by a set of components (molecules/neurons), a set of interactions (reactions/connections), and a description of the behavior (dynamics/firing rule). In contrast to chemical computing, an ANN is usually accompanied by a learning procedure. However, learning can be added to chemical computing by means of evolutionary computation [31, 32] or by transferring learning techniques from computational intelligence, e.g., Hebbian learning. In particular, this should be straight forward for explicitly defined chemical systems operating macroscopically, which are quite similar to dynamical neural networks (see Sec. 5.2 or ref. [14]).

But there are some remarkable differences: When we consider a reaction system with implicitly defined molecules and rules like the prime number chemistry in Sec. 5.1, we can easily obtain giant networks that are "invisible". When we look inside a reaction vessel, no component that represents a connection can be seen. Even the nodes of the chemical network cannot be easily identified because they are not spatially differentiated from each other, since a chemical node may be represented by a collection of molecules that are instances of one molecular species. The prime number chemistry is an example where a couple of simple rules imply a giant network, much larger than a human brain, e.g., for $n = 10^{30}$.

Another important difference to ANNs should be mentioned: When executing a chemical computation, only a subnetwork is active at a certain point in time, which is illustrated by Fig. 1. Since the size of a reaction vessel is limited, it can only contain a fraction of molecules from the set of all possible molecules. These present molecules together with all reactions that can occur among them can be regarded as the active reaction network. Due to internal or external dynamics, the set of molecular species in the reaction vessel can change, and thus the

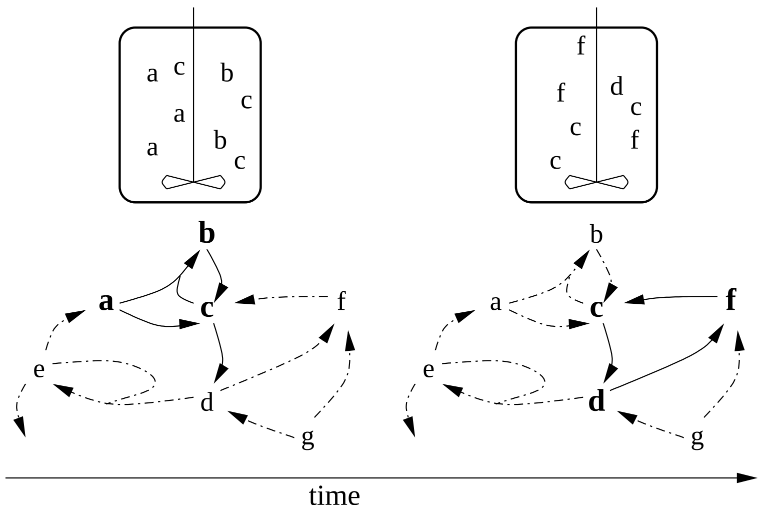

Fig. 1. Illustration of the invisible network, whose active part changes over time. A bold character denotes a molecular species that is present in the reactor. These species imply the currently active network highlighted by solid arrows. Note that a character in the reaction network denotes a molecular species, whereas the same character in the sketched reaction vessel denotes a concrete molecule (or instance) of that species.

active network evolves over time, too (Fig. 1). This phenomenon is captured theoretically by a movement through the set of chemical organizations [33, 30].

3 Chemical Programming

Programming a chemical computer means to define a chemical system, which is often also referred to as a *reaction system* or an *artificial chemistry* [34]. There are two fundamentally different approaches to chemical programming: (1) automatic programming by optimization, e.g. by means of evolutionary computation [31, 32], and (2) engineering by a human programmer, e.g. [1, 25]. Both approaches require specifying the following three aspects of the target chemistry:

(1) *Molecules*: In the first step, we have to specify how the molecules should look like. Should they be symbols or should they posses a structure, e.g., a sequence of characters [35], a hierarchical expression [36], or a graph like structure [37]. If molecules posses a structure, the definition of the reaction rules and the dynamics can refer to this structure, which allows to define large (even infinite) reaction systems, as exemplified by the prime number chemistry in Sec. 5.1. If the molecules are symbols, we have to specify the set of possible molecules *explicitly* by enumeration of all possible molecules, e.g., $\mathcal{M} = \{a, b, c, d\}$. If molecules posses a structure, we can define the set of all possible molecules implicitly, e.g., $\mathcal{M} = \{1, 2, \ldots, 10000\}$.

(2) *Reactions*: In the next step, we have to specify what happens when molecules collide. Real molecules can collide elastically or they can collide causing a reaction, which transforms the molecules. In a simple abstraction, a reaction rule is just a pair of two multisets of molecules, which specifies what kind of molecules can be transformed and replaced by what kind of molecules, e.g., a well known reaction rules is $(\{H_2, H_2, O_2\}, \{H_2O, H_2O\})$, which is written in chemical notation equivalently as $2H_2 + O_2 \rightarrow 2H_2O$. In general, reaction rules can become more complicated and can carry further information, such as parameters specifying kinetic constants or environmental conditions under which this reaction can occur. Analogously to molecules, reaction rules can be specified *explicitly* like $2H_2+O_2 \rightarrow 2H_2O$, or *implicitly*, as in the prime number chemistry.

(3) *Dynamics*: Finally, we have to specify the dynamics, which includes the geometry of reaction vessel. Do we assume a well-stirred vessel or vessel with a spatial structure? How do molecules collide? How are the reaction rules applied, e.g., deterministically or stochastically? Well-stirred deterministic reaction systems are usually simulated by integrating ordinary differential equations. Stochastic systems can be simulated by explicit stochastic collisions of individual molecules or by more advanced discrete event based methods like the Gillespie algorithm [38]. Spatial structures are usually introduced by some sort of cellular automata (e.g., lattice molecular automata [39]) or by compartments like in amorphous computing [25], membranes in P-systems [16] or mobile process calculi [40], or topology in MGS [17].

3.1 Strategies for Chemical Programming

We distinguish different strategies of chemical programming according to the components a programmer can manipulate:

(1) *Define molecules and reactions explicitly*: The programmer has to specify the set of molecules as a set of symbols and the reaction rules as a set of explicit transformation rules. An example for this approach is the hypercyclic memory (Sec. 5.2), the metabolic robot [41], and evolved chemical systems like those reported by Ziegler and Banzhaf [31].

(2) *Define molecules and reactions implicitly*: The programmer specifies molecules and reaction rules implicitly like demonstrated by the prime number chemistry in Sec. 5.1. For defining reaction rules implicitly, the molecules can not be just a list of symbols, rather they must posses a structure to which the definition of the reaction rules can refer to (see Sec. 3(1)). This approach is quite general, but because of its generality additional principle for guiding the programmer are required.

(3) *Change-molecules-only principle*: Again, the reaction rules are implicitly defined but fixed (predefined) and cannot be changed by the programmer. The programmer or an evolutionary process has “just” to select the right molecules and the dynamics, including the topology of the reaction space. This resembles the way real chemical computers are programmed, e.g., selecting appropriate DNA strands for solving a Hamiltonian path problem as in the famous example by Adleman [12]. Although the programmer has limited choices (compared to

the previous setting), the expressive power is the same, if a universal chemistry is used. A *universal chemistry* is defined as a chemistry that includes every possible (let's say, finite) reaction network topology. Such chemistry can, for example, easily be defined based on lambda-calculus [36] or combinators [42]. However, these abstract formalisms stemming from theoretical computer science can not be easily and intuitively handled by programmers, other approaches are more feasible for practical application (see Banâtre, Fradet, and Radenac in this volume).

(4) *Multi-level chemical programming*: As before, the programmer selects appropriate molecules and dynamics. At the same time, the "physics" can be manipulated, too, but at a slower rate. For example, the calculus specifying implicitly the set of possible molecules and the set of possible reactions can be altered and extended. By this, the function (meaning) of molecules can become more transparent (syntactic sugar). On a higher level of abstraction, molecules may be assembled to higher clusters resembling macro molecules or modules, which can again serve as building blocks for implicit definitions of other molecules, reaction rules, and dynamics. Therefore, a programmer operates on different levels, such as: Level 0: manipulation of the physics, e.g., combinator rules. Level 1: selecting (defining) the right molecules and reaction rules, and dynamics in the context of the chosen physics, e.g., selecting appropriate combinators. Level 2: Specifying higher level clusters, modules, macro-molecular complexes, e.g., based on membrane computing concepts.

4 Conclusion and Challenges

This essay discussed several aspects of artificial chemical computing. It has been shown that chemical-like systems possess a number of interesting properties, which appear especially feasible in domains like distributed computing, ambient computing, or organic computing. Furthermore, a couple of application scenarios have been described, including a first successful commercial application [23]. Taking heed of these facts, the chemical metaphor appears as a paradigm, which will qualitatively enrich our repertoire of programming techniques.

The road map of chemical computing includes a series of challenges: (1) Efficiency: How to obtain runtime and memory efficient chemical programs and their execution on electronic hardware? (2) Scalability: How do chemical computing paradigms scale up? (3) Programmability: How to program a chemical computer? (4) Adaptability and robustness: How to achieve self-adapting, learning, and reliable chemical computing systems? (5) Theory: How to describe chemical computing processes theoretically? Here, chemical organization theory [30] appears as a promising approach, especially when dealing with constructive chemical systems. Other sources for a theoretical base are classical dynamical systems theory and approaches from computer science like rewriting calculi [1, 2, 16, 40] and temporal logic. Furthermore we may investigate fundamental question concerning the power and limits of chemical computing by questions like: Can the chemical metaphor lead to new computational systems with abilities superior to conventional approaches, or even to systems that can not be realized by conventional approaches?

It is evident that the future will witness further integration of concepts from the natural sciences and computer science, which will reduce the differences between the living and the technological world. Like living systems, computing systems in the future will consist of decentralized and highly distributed components that interact with increasing autonomy and flexibility. For harnessing their potential it will be crucial to obtain new, organic methods for their construction and control. Chemical computing, which has been employed by nature with great success, offers a promising paradigm.

5 Appendix: Examples

5.1 Prime Number Chemistry

Banâtre and Le Metayer [1] suggested the numerical devision operator as an implicit reaction mechanism, which results in a prime number generating chemistry defined as follows (see ref. [41, 43] for details): the set of all possible molecules are all integers greater one and smaller $n+1$: $\mathcal{M} = \{2, 3, 4, \ldots, n\}$. The reaction rules are defined by a devision operation: $\mathcal{R} = \{a + b \to a + c \mid a, b, c \in \mathcal{M}, c = a/b,\ a \mod b = 0\} = \{4 + 2 \to 2 + 2,\ 6 + 2 \to 3 + 2,\ 6 + 3 \to 2 + 3,\ \ldots\}$. So, two molecules a and b can react, if a is a multiple of b. For the dynamics, we assume a well-stirred reaction vessel. The state of the reaction vessel of size M is represented by a vector (or equivalently by a multi-set) $P = (p_1, p_2, \ldots, p_M)$ where $p_i \in \mathcal{M}$. The dynamics is simulated by the following stochastic algorithm: (1) chose two integers $i, j \in \{1, \ldots, M\}, i \neq j$ randomly. (2) if there is a rule in $\mathcal{R}$ where $p_i + p_j$ matches the left hand side, replace p_i and p_j by the right hand side. (3) goto 1.

Assume that we initialize the reaction vessel P such that every molecule from $\mathcal{M}$ is contained in P, then we will surely reach a stationary state where all molecules from P are prime numbers and every prime number greater one and less or equal n is contained in P. The outcome (prime numbers present in P) is deterministic and in particular independent from the sequence of reactions, where the actual concentration of each prime number can vary and depends on the sequence of reactions. Now assume that P is smaller than $\mathcal{M}$, e.g., $M = 100$ and $n = 10000$. The outcome (molecular species present in P) is not deterministic. It depends on the sequence of updates, e.g., $P = (20, 24, 600)$ can result in the stable solutions $P = (20, 24, 30)$ or $P = (20, 24, 25)$. Note that the behavior (ability to produce prime numbers) depends critically on the reactor size M [41, 43].

5.2 Hypercyclic Associative Memory

Assume that we have an unreliable media, where all molecules decay sooner or later. In order to store data over a longer period, molecules have to be reproduced. Simple self-replicating molecules are not sufficient, since, as discussed by

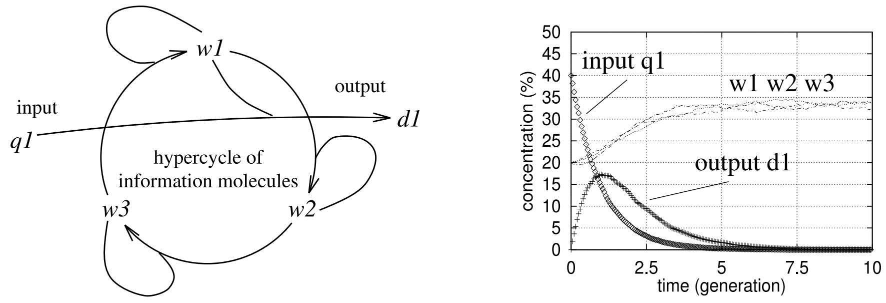

Fig. 2. Hypercyclic associative memory. **Left:** Illustration of the reaction network. An arrow represents a catalytic interaction where both reactants act as catalysts and are not used up. Only the active network is shown. **Right:** Example of a stochastic simulation of a query. 400 molecules of type q_1 are inserted. Reactor size $M = 1000$.

Eigen and Schuster [44], in a limited volume, self-replicating molecules compete for resources and can not coexist stably (exponential growth and no interaction assumed).

In the following example [41], three "units" of data $\{d_1, d_2, d_3\}$ are stored in three different molecules $\{w_1, w_2, w_3\}$. In order to query the memory, there are three input molecules $\{q_1, q_2, q_3\}$. Our demanded specifications are: the chemical system should store the data for a long period of time, under constant dilution of the molecules. The system should produce d_i provided q_i as input. We assume the following reactions: $\mathcal{R} = \{w_1 + w_2 \to w_1 + w_2 + w_2, w_2 + w_3 \to w_2 + w_3 + w_3, w_3 + w_1 \to w_1 + w_2 + w_1, w_1 + q_1 \to w_1 + q_1 + d_1, w_2 + q_2 \to w_1 + q_2 + d_2, w_3 + q_3 \to w_1 + q_3 + d_3\}$. For the dynamics, we assume a well-stirred reaction vessel that contains a constant number of M molecules. The state of the vessel is represented by a vector (or equivalently by a multi-set) $P = (p_1, p_2, \ldots, p_M)$ where $p_i \in \mathcal{M}$. The dynamics is simulated by the following stochastic algorithm: (1) chose three integers $i, j, k \in \{1, \ldots, M\}, i \neq j$ randomly. (2) if there is a rule $p_i + p_j \to p_i + p_j + x$ in $\mathcal{R}$, replace molecule p_k by x. (3) goto 1. This kind of stochastic algorithm is equivalent to the deterministic replicator equation and catalytic network equation. It is also used in several other works [45, 35, 46].

Figure 2 shows an example of a simulation where 400 molecules of type q_1 are inserted into a reactor that contains approximately the same amount of each information molecule $\{w_1, w_2, w_3\}$. Interaction of q_1 with w_1 results in the production of d_1. Since all molecules are subject to a dilution flow and q_1 is not produced, q_1 and d_1 are washed out while the concentrations of $\{w_1, w_2, w_3\}$ stabilize again.

Acknowledgment: I am grateful to F. Centler, N. Matsumaru, and K.-P. Zauner for helpful comments. This work was supported by the Federal Ministry of Education and Research (BMBF) Grant 0312704A to Friedrich Schiller University Jena.

References

1. Banâtre, J.P., Métayer, D.L.: The GAMMA model and its discipline of programming. Sci. Comput. Program. **15** (1990) 55–77
2. Berry, G., Boudol, G.: The chemical abstract machine. Theor. Comput. Sci. **96** (1992) 217–248
3. Adamatzky, A.: Universal dynamical computation in multidimensional excitable lattices. Int. J. Theor. Phys. **37** (1998) 3069–3108
4. Dörner, D.: Bauplan einer Seele. Rowohlt, Reinbeck (1999)
5. Brooks, R.A.: Coherent behavior from many adaptive processes. In Cliff, D., Husbands, P., Meyer, J.A., Wilson, S., eds.: From animals to animats 3, Cambridge, MA, MIT Press (1994) 22–29
6. Conrad, M.: Information processing in molecular systems. Currents in Modern Biology **5** (1972) 1–14
7. Liberman, E.A.: Cell as a molecular computer (MCC). Biofizika **17** (1972) 932–43
8. Liberman, E.A.: Analog-digital molecular cell. BioSytems **11** (1979) 111–24
9. Zauner, K.P.: Molecular information technology. Cr. Rev. Sol. State **30** (2005) 33–69
10. Tilden, W.A.: Introduction to the Study of Chemical Philosophy. 6 edn. Longmans, Green and Co., London (1888)
11. Küppers, B.O.: Information and the Origin of Life. MIT Press, Cambridge, MA (1990)
12. Adleman, L.M.: Molecular computation of solutions to combinatorical problems. Science **266** (1994) 1021
13. Conrad, M., Zauner, K.P.: Conformation-driven computing: A comparison of designs based on DNA, RNA, and protein. Supramol. Sci. **5** (1998) 787–790
14. Hjelmfelt, A., Weinberger, E.D., Ross, J.: Chemical implementation of neural networks and turing machines. Proc. Natl. Acad. Sci. USA **88** (1991) 10983–10987
15. Bazhenov, V.Y., Soskin, M.S., Taranenko, V.B., Vasnetsov, M.V.: Biopolymers for real-time optical processing. In Arsenault, H.H., ed.: Optical Processing and Computing, San Diego, Academic Press (1989) 103–44
16. Păun, G.: Computing with membranes. J. Comput. Syst. Sci. **61** (2000) 108–143
17. Giavitto, J.L., Michel, O.: MGS: a rule-based programming language for complex objects and collections. In van den Brand, M., Verma, R., eds.: Electronic Notes in Theoretical Computer Science. Volume 59., Elsevier Science Publishers (2001)
18. Laing, R.: Artificial organisms and autonomous cell rules. J. Cybernetics **2** (1972) 38–49
19. Suzuki, H., Ono, N., Yuta, K.: Several necessary conditions for the evolution of complex forms of life in an artificial environment. Artif. Life **9** (2003) 153–174
20. Banâtre, J.P., Métayer, D.L.: A new computational model and its discipline of programming. technical report RR-0566, INRIA (1986)
21. Suzuki, Y., Tanaka, H.: Symbolic chemical system based on abstract rewriting and its behavior pattern. Artif. Life and Robotics **1** (1997) 211–219
22. Suzuki, Y., Tsumoto, S., Tanaka, H.: Analysis of cycles in symbolic chemical system based on abstract rewriting system on multisets. In Langton, C.G., Shimohara, K., eds.: Artificial Life V, Cambridge, MA, MIT Press (1996) 521–528
23. Cliff, D., Grand, S.: The creatures global digital ecosystem. Artif. Life **5** (1999) 77–94
24. Astor, J.C., Adami, C.: A developmental model for the evolution of artificial neural networks. Artif. Life **6** (2000) 189–218

25. Abelson, H., Allen, D., Coore, D., Hanson, C., Homsy, G., Knight, T.F., Nagpal, R., Rauch, E., Sussman, G.J., Weiss, R., Homsy, G.: Amorphous computing. Commun. ACM **43** (2000) 74–82
26. Husbands, P., Smith, T., Jakobi, N., O'Shea, M.: Better living through chemistry: Evolving gasnets for robot control. Connect. Sci. **10** (1998) 185–210
27. Seelig, L.A., Rössler, O.E.: A chemical reaction flip-flop with one unique switching input. Zeitschrift für Naturforschung **27b** (1972) 1441–1444
28. Barrett, C.L., Mortveit, H.S., Reidys, C.M.: Elements of a theory of simulation II: sequential dynamical systems. Appl. Math. Comput. **107** (2000) 121–136
29. Reidys, C.M.: On acyclic orientations and sequential dynamical systems. Adv. Appl. Math. **27** (2001) 790–804
30. Dittrich, P., di Fenizio, P.S.: Chemical organization theory: Towards a theory of constructive dynamical systems. (submitted), preprint arXiv:q-bio.MN/0501016 **x** (2005) 1–7
31. Ziegler, J., Banzhaf, W.: Evolving control metabolisms for a robot. Artif. Life **7** (2001) 171 – 190
32. Bedau, M.A., Buchanan, A., Gazzola, G., Hanczyc, M., Maeke, T., McCaskill, J., Poli, I., Packard, N.H.: Evolutionary design of a DDPD model of ligation. In: 7th Int. Conf. on Artificial Evolution. LNCS, Springer, Berlin (2005) (in press)
33. Speroni Di Fenizio, P., Dittrich, P.: Artificial chemistry's global dynamics. movement in the lattice of organisation. The Journal of Three Dimensional Images **16** (2002) 160–163
34. Dittrich, P., Ziegler, J., Banzhaf, W.: Artificial chemistries - a review. Artif. Life **7** (2001) 225–275
35. Banzhaf, W.: Self-replicating sequences of binary numbers – foundations I and II: General and strings of length n = 4. Biol. Cybern. **69** (1993) 269–281
36. Fontana, W., Buss, L.W.: 'The arrival of the fittest': Toward a theory of biological organization. Bull. Math. Biol. **56** (1994) 1–64
37. Benkö, G., Flamm, C., Stadler, P.F.: A graph-based toy model of chemistry. J. Chem. Inf. Comput. Sci. **43** (2003) 2759–2767
38. Gillespie, D.T.: Exact stochastic simulation of coupled chemical-reactions. J. Phys. Chem. **81** (1977) 2340–2361
39. Mayer, B., Rasmussen, S.: Dynamics and simulation of micellar self-reproduction. Int. J. Mod. Phys. C **11** (2000) 809–826
40. Cardelli, L.: Brane calculi. In Danos, V., Schachter, V., eds.: Computational Methods in Systems Biology (CMSB 2004). Volume 3082 of LNCS., Berlin, Springer (2005) 257–278
41. Dittrich, P.: Selbstorganisation in einem System von Binärstrings mit algorithmischen Sekundärstrukturen. Diploma thesis, Dept. of Computer Science, University of Dortmund (1995)
42. Speroni di Fenizio, P.: A less abstract artficial chemistry. In Bedau, M.A., McCaskill, J.S., Packard, N.H., Rasmussen, S., eds.: Artificial Life VII, Cambridge, MA, MIT Press (2000) 49–53
43. Banzhaf, W., Dittrich, P., Rauhe, H.: Emergent computation by catalytic reactions. Nanotechnology **7** (1996) 307–314
44. Eigen, M., Schuster, P.: The hypercycle: a principle of natural self-organisation, part A. Naturwissenschaften **64** (1977) 541–565
45. Fontana, W., Wagner, G., Buss, L.W.: Beyond digital naturalism. Artif. Life **1/2** (1994) 211–227
46. Dittrich, P., Banzhaf, W.: Self-evolution in a constructive binary string system. Artif. Life **4** (1998) 203–220