The Bio-Chemical Information Processing Metaphor as a Programming Paradigm for Organic Computing

Peter Dittrich, Bio Systems Analysis Group, Jena Centre for Bioinformatics, and Department of Computer Science, Friedrich Schiller University Jena, D-07743 Jena, Germany

Abstract

All known life forms process information on a molecular level. This kind of chemical information processing is known to be robust, self-organizing, adaptive, decentralized, asynchronous, fault-tolerant, and evolvable. This paper 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 orchestrated interplay of many decentralized relatively simple components called molecules. Chemical programming encompassed then the definition of molecules, reaction rules, and the topology and dynamics of the reaction vessel. Through the nature of emergence, chemical computation is difficult to predict from the underlying microscopic interaction rules. Therfore, one central challenge, namely, how to program emergent chemical processes, is discussed in more detail.

1 Introduction

Organic computing takes inspiration from the living world in order to cope with the fast-growing complexity of technical information processing systems surrounding us [1, 2, 3, 4]. Information in biological systems is processed in a fundamentally different way than by today's mass-produced computing systems [5]. All known life forms process information using chemical processes [6]. Signal processing in bacteria, e.g., chemotaxis, or control of gene expression are well-known examples. 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).

The first part of this contribution provides a short introduction into chemical information processing and how the metaphor of chemistry can be employed to build technical information processing systems. The second part discusses the programmability of chemical computing architectures. It is argued that programming in a chemical language requires fundamentally new tools and a different way of thinking. This contribution focuses on what might be called "artificial chemical computing" in accordance to the term "artificial neural network" and as opposed to computing with real molecules.

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 (= physical change of a system).

Chemistry looks at the macro and micro level: On the macro level the 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 should 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.

Three aspects why chemistry is interesting for computer scientists should be mentioned. First, the way of explaining macroscopic phenomena by microscopic interactions of relatively simple compounds. It is quite appealing, when computation appears as an emergent process, where only the microscopic rules have to be specified and information processing appears as a global behavior, which might have some intersting properties, such as fine grained parallelism without central control. Second, chemical processes possess characteristic properties, which allow to implement particular systems like emotional systems [7] more naturally. Third, chemical processes themselves can be seen as a natural media for information processing either in vitro or in vivo [8, 9].

1.2 How are Chemical Explanations Organized?

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[1]. Second, we learn, how substances interact, in particular, we describe the outcome that results from their union. Reactive in-

[1] A chemical element is a compound that consists only of one atom type.

teractions 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 "Chlorine is not healthy in large quantities" we describe the property of chlorine by how it interacts with molecules in an organism. In fact, in chemistry, substances are often classified according to their reactive behavior.

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 [6].
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, planning, 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, e.g., a short path from an ant's nest to a food source.

1.4 Where Chemistry helps Computing ...

First, it should be noted that chemistry apparently helps computer science to build 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. This includes: constructing chemical-like formal system in order to model and master concurrent processes, e.g., GAMMA [11], CHAM [12]; 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 [13].
An early example, where the chemical metaphor appeared in computer science, are the *artificial molecular machines* suggested by Laing [14]. 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. [15] 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 Daniel Le Métayer, namely to introduce a new programming formalism that allows to automatize reasoning about programs, such as automatic semantic analysis [16, 11]. 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 [12] introduced the *chemical abstract machine* (CHAM) as a tool to model concurrent processes. Păun's P-Systems [17] stress the importance of membranes. Suzuki and Tanaka [18] introduced a rewriting system on multisets in order to study chemical systems, e.g., to investigate the properties of chemical cycles [19], and to model chemical-like systems including economic processes.
Within biological organisms, the endocrine system is a control system that transmits information by chemical messengers called hormones via a broadcast strategy. The humanoid robot torso COG [20] 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 [20]. In general, chemical-like systems can control the behavior and particularly emotions in artificial agents, e.g., the computer game Creatures [21] and the psychological model PSI by Dörner [7]. 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 [22]); and the control of amorphous computers [23]. Husbands et al. [24] introduced diffusing chemical substances in artificial neural networks (cf. GasNet).

2 Challenges

There are several challenges for future research in chemical computing: (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? 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? In the following, the problem of programmability is discussed in more detail.

2.1 How to Program a Chemical Computer?

Programming a chemical computer means to define a chemical system, which is often also referred to as a *reaction system* or an *artificial chemistry*. Since information processing emerges from the interaction of many simple components, programming a chemical systems appears to be difficult [25]. We have to rely mostly on our intuition when defining or programming a reaction system. In order to systematize the construction of an artificial chemical system, it is useful to partition its definition in the following three major steps:
(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 [26], a hierarchical expression [27], or a graph like structure [28]. 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 for example in the prime number chemistry [29] (Appendix). 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 carrying 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*, as in the previous example reaction rule, 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 simulated by explicit stochastic collisions of individual molecules or by more advanced discrete event based methods like the Gillespie algorithm [30]. Spatial structures are usually introduced by some sort of cellular automata (e.g., lattice molecular automata [31]) or by compartments like in amorphous computing [23], membranes in P-systems [32], or topology in MGS [33].

2.2 Strategies for Chemical Programming

We can 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, the metabolic robot [34], and evolved chemical systems like those reported by Ziegler and Banzhaf [35].
(2) *Define molecules and reactions implicitly*: The programmer can specify molecules and reaction rules implicitly like demonstrated by the prime number chemistry in Sec. 3. This approach is quite general, but because of its generality it does not guide the programmer.
(3) *Change molecules principle*: Again, the reaction rules are implicitly defined but fixed (predefined) and cannot be changed by the programmer. The programmer 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 stands for solving a Hamiltonian-Path as in the famous example by Adleman [36]. 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 [27] or combinators [37]. Because such abstract formalisms from theoretical computer science can not be easily and intuitively handled by programmers, other approaches are more feasible for practical application [38].
(4) *Multi-level chemical programming*: As before, the programmer selects appropriate molecules and dynamics. At the same time, the "physics" can be manipulated, too. 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.

3 Conclusion

Taking bio-chemical information processing as an inspiration for organic computing appears to be attractive, since chemical systems possess a number of desirable properties. Various approaches following this line can already been found. However, chemical programming will surely not replace our current methods, e.g., implementing a word-processor on a chemical basis is not feasible. It is more likely that artificial chemistries will be integrated as subsystems together with other higher-level computing concepts. Still, comprehensive techniques for programming chemical-like technical systems are missing and systematic qualitative and quantitative studies concerning the pro and cons have to be carried out. However, taking heed of the different approaches envisioned here, advances in chemical programming can be expected in the near future.

Appendix: Prime Number Chemistry

Banâtre and Le Metayer [11] suggested the numerical devision operator as an implicit reaction mechanism, which results in a prime number generating chemistry defined as follows: 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: $\mathcal{R} = \{a + b \rightarrow a + c \mid a, b, c \in \mathcal{M},\, c = a/b,\, a \mod b = 0\} = \{4 + 2 \rightarrow 2 + 2,\, 6 + 2 \rightarrow 3 + 2,\, 6 + 3 \rightarrow 2 + 3,\, \ldots\}$. 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 (see ref. [29] for details).

Acknowledgment

This work was supported by the *Federal Ministry of Education and Research* (BMBF) Grant 0312704A to *Friedrich Schiller University Jena*.

4 References

[1] Organic computing. URL: www.organic-computing.org, visited: 22.01.2005, last updated: 07.06.2005 (2004)

[2] Müller-Schloer, C., Malsburg, M., Würtz, R.P.: Aktuelles Schlagwort: Organic Computing. Informatik Spektrum **27** (2004) 332–336

[3] von der Malsburg, C.: The challenge of organic computing (1999) Memorandum, Computer Science Department.

[4] Würtz, R.P.: Organic computing for face and object recognition. In Dadam, P., Reichert, M., eds.: Informatik 2004. Volume 2., Gesellschaft für Informatik (2004) 636–640

[5] Paton, R.C., ed.: Computing with Biological Metaphors. Chapman and Hall, London (1994)

[6] Küppers, B.O.: Information and the Origin of Life. MIT Press, Cambridge, MA (1990)

[7] Dörner, D.: Bauplan einer Seele. Rowohlt, Reinbeck (1999)

[8] Conrad, M.: Information processing in molecular systems. Currents in Modern Biology **5** (1972) 1–14

[9] Liberman, E.A.: Cell as a molecular computer (MCC). Biofizika **17** (1972) 932–43

[10] Tilden, W.A.: Introduction to the Study of Chemical Philosophy. 6 edn. Longmans, Green and Co., London (1888)

[11] Banâtre, J.P., Métayer, D.L.: The GAMMA model and its discipline of programming. Sci. Comput. Program. **15** (1990) 55–77

[12] Berry, G., Boudol, G.: The chemical abstract machine. Theor. Comput. Sci. **96** (1992) 217–248

[13] Adamatzky, A.: Universal dynamical computation in multidimensional excitable lattices. Int. J. Theor. Phys. **37** (1998) 3069–3108

[14] Laing, R.: Artificial organisms and autonomous cell rules. J. Cybernetics **2** (1972) 38–49

[15] 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

[16] Banâtre, J.P., Métayer, D.L.: A new computational model and its discipline of programming. technical report RR-0566, INRIA (1986)

[17] Păun, G.: Computing with membranes. J. Comput. Syst. Sci. **61** (2000) 108–143

[18] Suzuki, Y., Tanaka, H.: Symbolic chemical system based on abstract rewriting and its behavior pattern. Artif. Life and Robotics **1** (1997) 211–219

[19] 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
[20] 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
[21] Cliff, D., Grand, S.: The creatures global digital ecosystem. Artif. Life **5** (1999) 77–94
[22] Astor, J.C., Adami, C.: A developmental model for the evolution of artificial neural networks. Artif. Life **6** (2000) 189–218
[23] 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
[24] Husbands, P., Smith, T., Jakobi, N., O'Shea, M.: Better living through chemistry: Evolving gasnets for robot control. Connect. Sci. **10** (1998) 185–210
[25] Conrad, M.: The price of programmability. In Herken, R., ed.: The Universal Turing Machine: A Fifty Year Survey. Oxford University Press, New York (1988) 285–307
[26] 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
[27] Fontana, W., Buss, L.W.: 'The arrival of the fittest': Toward a theory of biological organization. Bull. Math. Biol. **56** (1994) 1–64
[28] Benkö, G., Flamm, C., Stadler, P.F.: A graph-based toy model of chemistry. J. Chem. Inf. Comput. Sci. **43** (2003) 2759–2767
[29] Banzhaf, W., Dittrich, P., Rauhe, H.: Emergent computation by catalytic reactions. Nanotechnology **7** (1996) 307–314
[30] Gillespie, D.T.: Exact stochastic simulation of coupled chemical-reactions. J. Phys. Chem. **81** (1977) 2340–2361
[31] Mayer, B., Rasmussen, S.: Dynamics and simulation of micellar self-reproduction. Int. J. Mod. Phys. C **11** (2000) 809–826
[32] Păun, G.: Computing with membranes. Journal of Computer and System Sciences **61** (2000) 108–143
[33] 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)
[34] Dittrich, P.: Selbstorganisation in einem System von Binärstrings mit algorithmischen Sekundärstrukturen. Diploma thesis, Dept. of Computer Science, Univ. of Dortmund, Dortmund (1995)
[35] Ziegler, J., Banzhaf, W.: Evolving control metabolisms for a robot. Artif. Life **7** (2001) 171 – 190
[36] Adleman, L.M.: Molecular computation of solutions to combinatorical problems. Science **266** (1994) 1021
[37] 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
[38] Banâtre, J.P., Fradet, P., Radenac, Y.: Principles of chemical programming. In Abdennadher, S., Ringeissen, C., eds.: RULE'04 Fifth International Workshop on Rule-Based Programming, Technical Report AIB-2004-04, Dept. of Computer Science, RWTH Aachen, Germany (2004) 98–108