

Outlining an Unconventional, Adaptive, and Particle-Based Reconfigurable Computer Architecture

Christof Teuscher

University of California, San Diego (UCSD), Department of Cognitive Science,
9500 Gilman Drive, La Jolla, CA 92093-0515, USA
christof@teuscher.ch
www.teuscher.ch/christof

Abstract. The quest for novel and unconventional computing machines is mainly motivated by the man-machine dichotomy and by the belief that dealing with new physical computing substrates, new environments, and new applications will require new paradigms to organize, train, program, and to interact with them. The goal of this contribution is to delineate a possible way to address the general scientific challenge of seeking for further progress and new metaphors in computer science by means of unconventional approaches. Here we outline an amalgamation of (1) a particle-based, randomly interconnected, and reconfigurable substrate, (2) membrane systems, and (3) artificial chemistries in combination with (4) an unconventional adaptation paradigm.

1 Introduction and Motivation

Biologically-inspired computing (see for example [28, 18] for a general introduction), also commonly called *natural computing*, is an interdisciplinary area of research which is heavily relied on the fields of biology, computer science, and mathematics. It is the study of computational systems that use ideas and draw inspiration from natural organisms to build large, adaptive, complex, and dynamical systems. The principal goal of biologically-inspired computing is to make machines more lifelike and to endow them with properties that traditional machines typically do not possess, such as for example adaptation, learning, evolution, growth, development, and fault-tolerance.

It is evident that biological organisms operate on completely different principles from those of computer science (i.e., the man-machine dichotomy). Whereas life itself might be defined as a chemical system capable of self-reproduction and of evolution, computers constitute a fundamentally different environment where such processes are not naturally occurring. This and our still poor understanding of nature in many aspects makes it particularly difficult to copy it. Further difficulty often resides in the way how information in computers is represented and processed. Mimicking biological information processing on computing devices, for example, which usually process information serially, is highly inefficient because of the massively parallel character of biological systems. Also, whereas

the concepts of *computability* and *universal computation* are undoubtedly central to theoretical computer science, their importance might be questioned with regards to biological organisms and biologically-inspired computing machines. The well known concept of computation (as defined for example by a Turing machine) can in most cases not straightforwardly be applied to biological components or entire organisms and there is no evidence that such a system can compute universally, on the contrary, nature usually prefers highly specialized units. For example, it is much debated among neuroscientists, cognitive scientists, computer scientists, and philosophers whether the metaphor of the brain or mind as a digital computer is reasonable. However, although we have experienced in the past that biological organisms are generally difficult to describe by algorithmic processes, there is little reason to believe that they compute beyond the algorithmic horizon [33], since, at the bottom, one might reason, it is all just physical stuff doing what it “must.” Yet another debate is focused on what cognitive paradigms should be used in order to obtain “intelligent” agents. This seems to somehow become an eternal and useless discussion since—as in many other fields—there is no best model. Different models are better for different things in different contexts.

The quest for novel computing machines and concepts is mainly motivated by the observation that fundamental progress in machine intelligence and artificial life seem to stagnate [3]. For example, one of the keys to machine intelligence is computers that learn in an open and unrestricted way, and we are still just scratching the surface of this problem. Connectionist models have been unable to faithfully model the nervous systems of even the simplest living things and parallel programming has failed to produce general methods for programming massively parallel systems. Our abilities to program complex systems are simply not keeping up with the desire to solve complex problems. But is “programming”—by means of a high-level language for example—really the best solution for this kind of challenge? I do have more hope in self-organization and learning when it comes to the creation of highly complex and massively parallel systems. But this opens numerous additional problems, such as the general lack of systematic and formal approaches which would allow to gradually create hierarchical and complex systems that scale-up well, for example.

Tomorrow’s grand challenges for computer science are not likely to be bound to any specific real-world application, but one would rather like to have general mechanisms—at least in a first step—that would allow to gradually and automatically create complex systems. Typical issues often mentioned are robustness towards failures, scalability, adaptation towards an ever-changing environment, and more complex machines in general. Also, tomorrow’s computers are likely to be ubiquitous and invisible, which requires a paradigm shift in how we organize, train, program, and interact with them. We believe that these challenges are best approached by unconventional paradigms instead of “straying” around the well-known concepts, changing the model’s parameters, and putting hope in increasing computing power. Using Brooks’ words, we rather believe in “[...] something fundamental and currently unimagined in our models” [3] than in all

other possibilities, although they might play a role as well. We need new tools and new concepts that move seamlessly between brain, cognition, and computation—a finding that has also been made in a 2002 NSF/DOC-sponsored report [27].

The general goal of this contribution—presented in the form of an outline—is to illustrate *one* possible way to address the general scientific challenge of seeking for further progress and new metaphors in computer science by means of unconventional methods. The presented ideas should be seen as a fully integrative approach for an unconventional computing machine, which includes everything from the underlying hardware to its organizational principles. The work is ongoing and further develops some of the ideas first presented in [32]. Please also note that the goal of this contribution is to give an overview and it is therefore not self-contained and does not address all the details.

The remainder of the paper is as following: Section 2 gives an overview on the architecture and mentions some relevant related work. Section 3 illustrates the programmable reactor multitude and its communication structure. The implementation of cells and hierarchical structures is outlined in Section 4 and Section 5 explains the basics of chemical blending. Finally, Section 6 concludes the paper and delineates future work.

2 Architectural Overview

Different areas in computer science on different levels are facing the same fundamental problems. For example, an ideal mobile phone communication network is self-organized, robust, self-healing, and scalable, which might simply be translated into the capacity to adapt to a complex, uncertain, and ever-changing environment. The same generally applies to distributed sensor networks, mesh networks, the internet, collective robotics, amorphous computers, molecular electronics, and many other fields. All these applications are typically characterized by decentralized control, asynchronous operation, and by the fact that failures may occur at any time. However, they differ significantly in several aspects, such as for example power consumption, resources available, agent density, and communication infrastructure.

The specific goal of this work was to propose a novel reconfigurable computer architecture that is based on an amalgamation of (1) a particle-based and randomly interconnected substrate, (2) membrane systems, and (3) artificial chemistries in combination with (4) an unconventional adaptation paradigm. An overview of the architecture is given in Figure 1.

In order to obtain a fault-tolerant system, the proposed architecture relies on a simple, irregular, inhomogeneous, locally interconnected, asynchronously operating, and imperfect particle-based substrate (see Section 3), not unlike an amorphous computer. Currently, the only possibility to build a perfect machine out of imperfect components is to make use of redundancy at some level, such as spare components, which clearly favors a particle-based implementation. Scalability is assured by avoiding central control and by using local interactions only. We will also make extensive use of artificial chemistries, which represent—if appro-

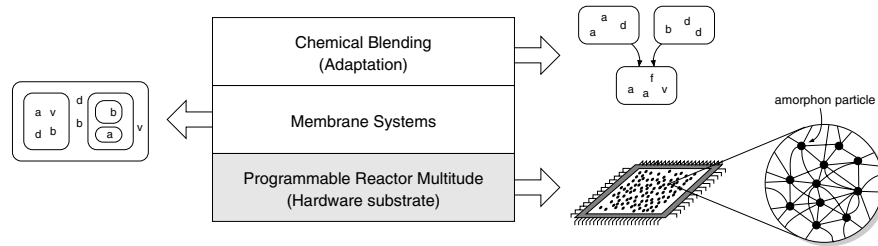


Fig. 1. Overview of the different levels of the proposed reconfigurable computer architecture: (1) particle substrate, (2) membrane systems used to build hierarchical levels, and (3) an unconventional method to search for “good” membrane systems. Artificial chemistries are used on all levels

priately used—an ideal means to compute in uncertain environments. Further, they have also been identified as potentially very promising for the perpetual creation of novelty [11], a feature that shall be later used to support adaptation. In order to be able to create hierarchical organizations, we will make use of cells and membranes throughout the system (see Section 4). Thereby, membrane systems will serve as a main source of inspiration. Finally, adaptation is achieved by a method inspired by *conceptual blending* [7,6], a framework of cognitive science that tries to explain how we think and deal with mental concepts. However, instead of dealing with concepts, we will use artificial chemistries in combination with membrane systems (see Section 5).

A number of completed and ongoing projects in the computer science research community are dealing with alternative computer architectures. We shall briefly focus on some relevant representatives. One of the most prominent projects in the field is certainly MIT’s *amorphous computing project*¹ [1, 19]. Amorphous computing is the development of organizational principles and programming languages for obtaining coherent global behavior from the local cooperation of myriads of unreliable parts that all contain the same program and that are interconnected in unknown, irregular, and time-varying ways. In biology, this question has been recognized as fundamental in the context of animals (such as ants, bees, etc.) that cooperate and form organizations. Amorphous computing brings the question “down” to computing science and engineering. Using the metaphor of biology, the cells cooperate to form a multicellular organism (also called *programmable multitude*) under the direction of a genetic program shared by the members of the colony.

*Blob Computing*² [15, 14] is an ongoing project that is based on a coupled language-machine approach. The goal is to present an alternative to the well-established von Neumann paradigm in order to tackle the major limitations of current computer architectures. The model is designed to exploit “space” and is based on a massively parallel and asynchronous computational architecture

¹ Website: <http://www.swiss.ai.mit.edu/projects/amorphous>

² Website: <http://blob.lri.fr>

that offers a good scalability combined with a language that allows to fully exploit the underlying hardware. The programming language relies on a virtual machine called *graph machine*, which is a self-modifying and self-developing net of automaton. This basics of the language date back to Gruau's work on cellular encoding (see for example [12, 13]). Each node (i.e., automaton) of the graph can locally modify the graph and can apply cell division instructions in order to develop the graph. A cell can for example simulate a simplified artificial neuron, thus a cellular code can develop and simulate an artificial neural network. Mapping, a central challenge of parallel programming, is the problem of determining which processor will simulate which cell of the graph. Naturally, a good placement reduces the distance between the nodes that communicate. Gruau *et al.* use physical forces for optimizing the shape and the location of a blob and thus solve the mapping problem in an elegant way. As a proof of concept, they illustrate several interesting examples and experiments in [14], such as QuickSort, building neural networks, and matrix multiplication.

Both, the amorphous computing and the Blob computing project are based on an irregular, randomly arranged, and locally interconnected underlying hardware structure. Examples of unconventional reconfigurable hardware architectures based on a regular arrangement of the basic components are the *Embryonics* [17], the *POetic* project [31, 35], and the *CellMatrix* architecture [16], which all resemble traditional *Field Programmable Gate Arrays* (FPGAs) [34]. Finally, on a totally different level, but with very similar goals (i.e., robustness, self-healing and repair, adaptation), IBM has launched the *autonomic computing*³ initiative some time ago.

3 The Programmable Reactor Multitude

At the bottom of our reconfigurable architecture (see Figure 1) lies the *Programmable Reactor Multitude* (PRM), which shall be briefly described in the next two sections. The programmable reactor multitude (i.e., the hardware) is made up of a set of interconnected particles, each of which contains as a main element a chemical reactor.

3.1 Communication Structure

MIT's amorphous computer communication model assumes that all processors have a circular broadcast of approximately the same fixed radius (large compared to the size of the processor) and share a single communication channel. Since our goal was to propose a new reconfigurable computer architecture, we have chosen wire-based communication instead of wireless broadcast. The communication model is illustrated in Figure 2. Each processor i receives inputs from an average of K_{avg} randomly chosen neighbors that lie within a maximum radius of c_i . Self-connections are not allowed, but multiple connections are allowed. Further, all

³ Website: <http://www.research.ibm.com/autonomic>

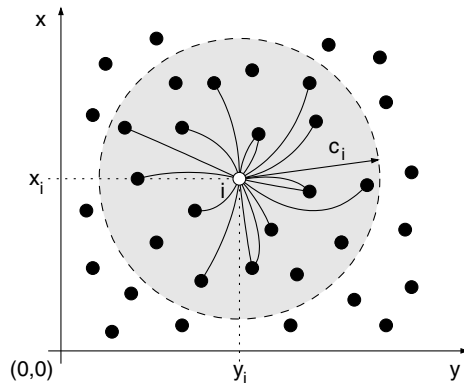


Fig. 2. Illustration of the inter-processor's communication model. Each randomly arranged processing element i (i.e., particle) receives inputs from an average of K_{avg} of its neighbors that lie within an maximum radius of c_i . Multiple connections to the same neighbors are allowed, self-connections are not allowed

communication channels are single and bidirectional. We also assume that the processors are randomly and densely distributed on a two-dimensional surface and that their physical position as well as their wiring are fixed, but that they might be unreliable. For applications such as reconfigurable silicon-based circuits, printable digital circuits, etc., this seems to be a reasonable constraint. The processors do not possess any information about their physical location and can only communicate with their immediate neighbors.

The main reason for choosing this alternative communication model was to facilitate circuit-based real-world implementations, where long-distance connections are generally costly in terms of resources used and where radio-broadcast is inadequate and unnecessary. It was further inspired by small-world graphs [30], which have short average interconnection lengths but high clustering coefficients. Although our connection graph does not exactly have small-world properties nor is it scale-free, it is a reasonably good approximation and easy to create. Obviously, a necessary condition for a correct operation is a connection graph without disconnected regions, as this would prevent communication among all particles. In order to provide a fault-tolerant communication structure, however, redundant links, i.e., multiple paths between two locations are necessary since it must be possible in such a case to re-route traffic via alternative routes.

The message routing protocol is very simple and is based on chemical gradients only. This makes the message handling easy in case of faulty connections and processors, in which case the messages simply choose an alternative route in the direction of the gradient. However, at the same time, the expected time to deliver a message is hard to estimate since its exact route is unknown. There are basically three types of communication primitives: (1) broadcast a message to all neighboring processors, (2) deliver a message to the source of a certain gradient, and (3) search for a processor that satisfies a specific condition. For more information, the interested reader is also referred to [32].

3.2 The Particle's Functionality

Several processor architectures have been proposed within the amorphous computing project at MIT. Here, we will focus on yet another version, which is mainly based on a chemical reactor paradigm, which helps to keep the overall design simple, uniform, and universal. The main part of the processor, also called *particle* or *amorphon* is essentially composed of a stochastic chemical reactor, which can host a limited number of molecules and reactions of an artificial chemistry. An *artificial chemistry* [5] is a man-made system defined by a set of reactions, a set of molecules, and by reaction dynamics which describe how the elements interact. Here, we use a rewriting-based chemistry with molecules such as for example $w = \{a, b, c\}$ and reactions such as $r = \{a \rightarrow c, b \rightarrow a\}$ (see also Section 4). In our case, the reaction dynamics are as following: the algorithm randomly draws a reaction and then checks whether it can be applied with the molecules currently present in the reactor. If yes, the rewriting takes place, otherwise a new reaction will be drawn.

Figure 3 shows a simplified view of a particle with its stochastic chemical reactor. All incoming messages containing molecules or reactions are fed into the reactor. Once in there, each chemical has a probability p_{leave} to leave the reactor and to be distributed to their interconnected neighbors according to the protocol they specify with their special symbols, which shall be described in the next section. We will see in Section 4 how the membranes will restrict the chemicals to float around on the entire substrate. The resulting system is a distributed reactor network with similar chemical concentrations in all reactors. The system is entirely reconfigurable since the contents of the reactors is not restricted.

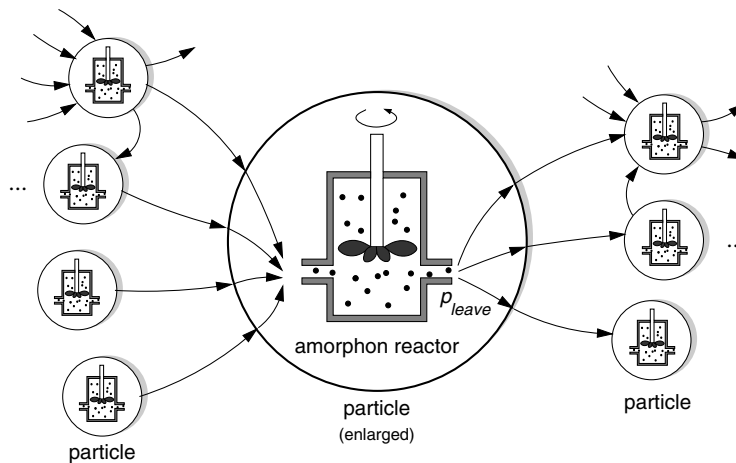


Fig. 3. Simplified view of a particle and its stochastic chemical reactor. Each chemical (i.e., molecule or reaction) has a probability of p_{leave} to leave the reactor. The particle is interconnected with its local neighbors as explained in the previous section

Note that this only gives a partial overview on the system. Each particle contains further components that deal for example with gradient information, that locally store their values, and that route messages. We shall not go into further details here as the chemical reactor represents the central element. Also, the issue on how the reactors are initialized is not addressed here and we simply assume a certain initial state of the entire programmable reactor multitude for the sake of simplicity.

In the next section we shall see how membrane systems can be built on top of a programmable reactor multitude.

4 Implementing Hierarchical Organizations with Membrane Systems

The biological cell is undoubtedly the structural and functional unit of all living organisms. Because of its importance, countless computational models with to goal to mimic or copy cells in nature have been proposed. In this section, we will draw inspiration from a particular cellular model in order to implement hierarchical membrane structures on top of a programmable reactor multitude. The main goal of this approach is to create hierarchical organizations, which will later be used by chemical blending (see Figure 1 and Section 5). Also, hierarchical composition is ubiquitous in any physical and biological system and offers many benefits. For example, it is a means to divide and “hide” complexity, to save resources since the building-blocks might be shared and re-used, and to create higher levels of abstraction. The formation of groups and hierarchies was also extensively addressed in the amorphous computing project. For more information, see for example [4, 20].

4.1 Membrane Systems

In 1998, Paun initiated *P systems* (or *membrane computing*) [22, 24] as a highly parallel, though theoretical computational model afar inspired by biochemistry and by some of the basic features of biological membranes. A typical P system (note that many variations exist) consists of cell-like membranes placed inside a unique “skin” membrane. Multisets of *symbol objects* and a set of *evolution rules* are then placed inside the regions delimited by the membranes. This artificial chemistry, which evolves over time, can then be used to compute. Figure 4 shows an example of a P system that generates n^2 , $n \geq 1$. For more details see [22]. The fact that all rules have to be applied in parallel in all membranes adds resources and difficulties for hardware implementations because of the global control signals needed for the synchronization. However, it makes the system easier to create and to analyze. Despite this, Petreska and Teuscher [26] have recently proposed a first P system implementation on traditional reconfigurable circuits, which greatly speeds up their simulation. An additional problem is the fact that membrane systems are usually engineered by hand as no methodology exists so far on how to set up an artificial chemistry for a given task. Specifying

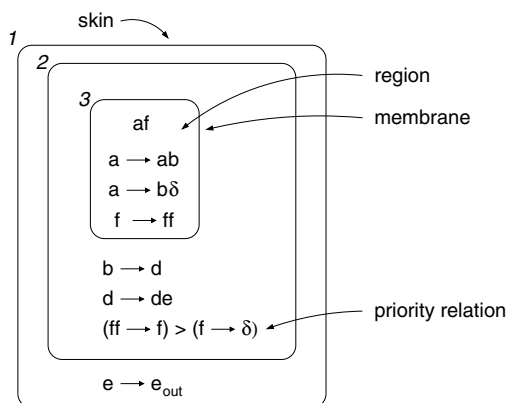


Fig. 4. A P system that generates n^2 , $n \geq 1$, where n is the number of steps *before* the first application of the rule $a \rightarrow b\delta$. Redrawn from [23]

a non-trivial task algorithmically might of course be very hard, but if the problem is solvable by a machine, we know it can be done. Nevertheless, P systems are particularly interesting for our purposes since they allow to easily create hierarchies, which we consider a key issue for the creation of complex systems. Also, this will allow us to create compartments for constraining the unlimited spreading of the chemicals in the programmable reactor multitude network.

4.2 Membrane Systems on a Programmable Reactor Multitude

In order to be able to efficiently and elegantly implement membrane systems on a programmable reactor multitude, we have modified classical P systems in several points. The following list provides an incomplete overview of the main points:

- The rules are no longer applied in parallel in each membrane, instead they are applied stochastically and asynchronously and there are no priority relations between the rules.
- Compared to classical P systems, where the rules do not usually change, our rules can be rewritten and created by rules as well. For instance, the rule $a(b \rightarrow c) \rightarrow d$ requires a molecule a and a rule $b \rightarrow c$ to be applied. The two elements would then be removed from the chemistry and replaced by d . This possibility allows to (self-) modify the reactor’s “program,” i.e., its rules, and offers more flexibility than a fixed set of rules.
- A set of special symbols defines additional actions for each rule: $r = (u, s, v) = (u \rightarrow sv)$, where u, v are multisets over the symbols of an alphabet V and the set R^* of all reactions r , and where s is a *special operator symbol* from a set S . The special symbols allow to send objects to different neighboring membranes, to dissolve and create membranes, and will later allow us to implement chemical blending, but overall, they offer simply a mechanism to attach a certain additional action of any kind to each rule. The implementation of the special symbols (i.e., the microsteps) is generally done outside the reactor.

- The inter-membrane communication is modified such that a membrane can not only send objects to its outer compartment, but also to a specific membrane in that compartment.

These modifications are less guided by theoretical than by practical considerations. Whether they make our membrane systems computationally more or less powerful than their classical counterparts remains to be investigated, but was not the focus on this work.

From an implementational point of view, every cell in a membrane system will be represented by a set of neighboring particles from the programmable reactor multitude. Chemicals of that cell can only leave this set of processors by means of special commands, otherwise they will move inside the cell only. Remember also that each reactor has a limited capacity, i.e., the more chemicals have to be stored, the more reactors are required to make up the cell. Also, the goal is to have a larger number of reactors available for each cell than it would be strictly necessary in order to obtain a fault-tolerant system. Ideally, there should be redundancy that allows to “lose” at least one particle out of one cell. In such a case, it is for instance possible to hold a concentration of chemicals (e.g, a) above a certain threshold by means of simple rules, even if the concentration is disturbed by removing or adding reactors. Figure 5 illustrates this with the following rules: $r_1 = a \rightarrow a^2$ and $r_2 = a^m \rightarrow a^n$, where $n < m$. Rule r_1 lets the concentration grow constantly but slowly, whereas rule r_2 reduces the number of molecules once it has reached a certain upper threshold m instantly by $m - n$

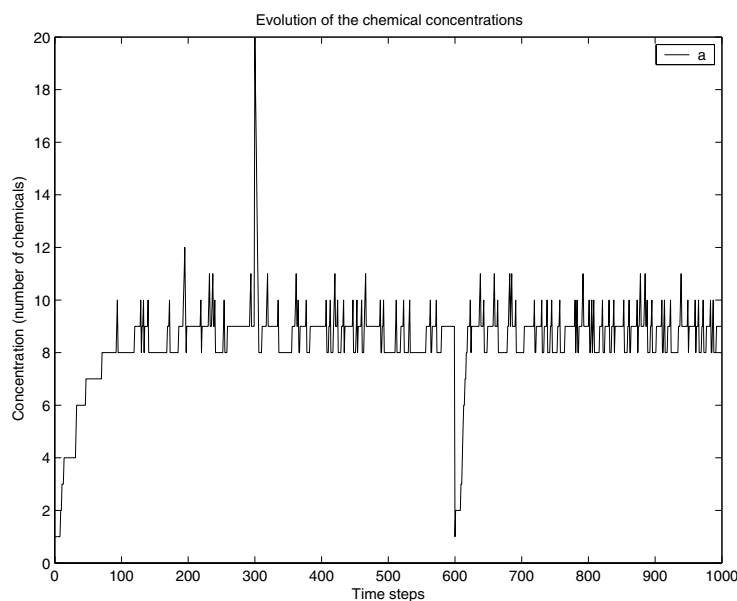


Fig. 5. Holding a concentration of chemicals a constant. The disturbances at time steps 300 and 600 do only briefly alter the concentration

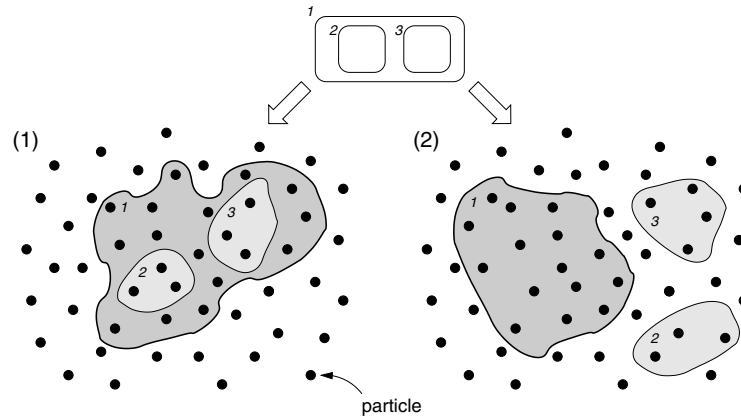


Fig. 6. Two possibilities of implementing hierarchical membrane systems on a programmable reactor multitude. See text for description

molecules. As one can see, the concentration shows a sawtooth-like behavior for $m = 10$ and $n = 8$, but proves to be very robust against unexpectedly external influences.

Let us now see how we can implement a membrane system, as shown in Figure 4, on a programmable reactor multitude. There are two principal ways: (1) respect the membrane's hierarchical organization or (2) lay the individual cells flat out. The two possibilities are illustrated in Figure 6. Possibility (1) seems more natural, but there is a major drawback when a new cell has to be added. In that case, existing cells would have to be enlarged in order to create room for the new cells, which represents a time-consuming and non trivial task—especially in the case of many hierarchical levels—given our simple processing elements. Possibility (2) avoids this drawback, but in that case the communication between the cells is more complicated. However, in most cases, this possibility seems to be more appropriate, since adding and removing cells is greatly simplified.

5 Chemical Blending

Conceptual blending (or *conceptual integration*) [7, 6] is a theory developed by Fauconnier and Turner about conceptual spaces and how they develop and proliferate as we talk and think. Conceptual spaces consist of elements and relations among them and are of course not directly instantiated in the brain, instead, they should be seen as a formalism invented by researchers to address and certain issues of their investigation. When two conceptual spaces are blended together, the new space contains parts of the original spaces, but it usually also contains emergent structure. Very simple examples are “houseboat” or “computer virus,” a more complex blend would be “digging one own’s grave.” For most real-world situations that are more complex than a simple metaphor, blends develop in larger *conceptual integration networks*, which are networks of conceptual spaces

and conceptual mappings. The blend's "quality" and "usefulness" is guided by a set of *optimality principles* (see [7] for more details), most of which use human judgment, which makes them hard to implement in computational frameworks. Since the entire blending framework lacks a formal approach, different people have worked on explicit computational approaches in recent years [36,8,9,25,10]. Also, conceptual spaces and blending are just a good tool to study meaning in natural language, metaphors, and concepts, but they are not generally suitable to talk about the structure of things [9]. Hence, Goguen and Harrell recently proposed a blending algorithm called *structural blending* [9,10], which also takes into account structure.

While all current computational approaches deal with concepts, we are interested in a very different approach here: we are looking for an unconventional adaptation paradigm in the context of artificial chemistries and membrane systems. Drawing inspiration from the constructing and optimality principles of blending seemed promising since blending creates emergent structure, i.e., novelty, and could therefore in principle be useful to discover new solutions. Also, as mentioned earlier, artificial chemistries have been identified as potentially very promising for the perpetual creation of novelty, which, together with a blending-inspired method, could lead to interesting properties. Finally, please note that the chemical blending approach is *not* intended to faithfully model blending, but only draws inspiration from it instead.

The following analogies were basically used: a mental space is replaced by a membrane system whereas objects and rules became molecules and reactions. Also, the three constructing principles of blending [7], namely (1) composition, (2) completion, and (3) elaboration can straightforwardly be replaced by composing, completing, and applying the rules of the membrane system. As the blending's optimality principles do basically only make sense for conceptual integration, we have completely replaced them in a first step by an alternative fitness-based measure as commonly used in evolutionary algorithms. The development of specialized optimality principles for artificial chemistries is envisaged for future work.

Figure 7 provides an overview on how a new cell is blended from two single-membrane input cells. We have implemented one possible method which shall now briefly be described. Let us assume two single-membrane systems (as shown in Figure 7) that contain a certain number of molecules and reactions. The goal is to create a new single-membrane system that contains "emergent content" from the two original cells. For the sake of simplicity and because the reactions represent the cell's "program," we will only focus on the chemical reactions and not on the molecules here. One of the first steps of blending consists in establishing a cross-space mapping between related elements. In order to do this, we introduce an *activity* and a *similarity measure*. The similarity measure considers the rule's symbols and structure and measures how related they are whereas the activity measure compares how often the rules are used. The idea is that rules with similar or opposite structure and activities should be more likely to be combined to form new rules than any other combination. The mapping is

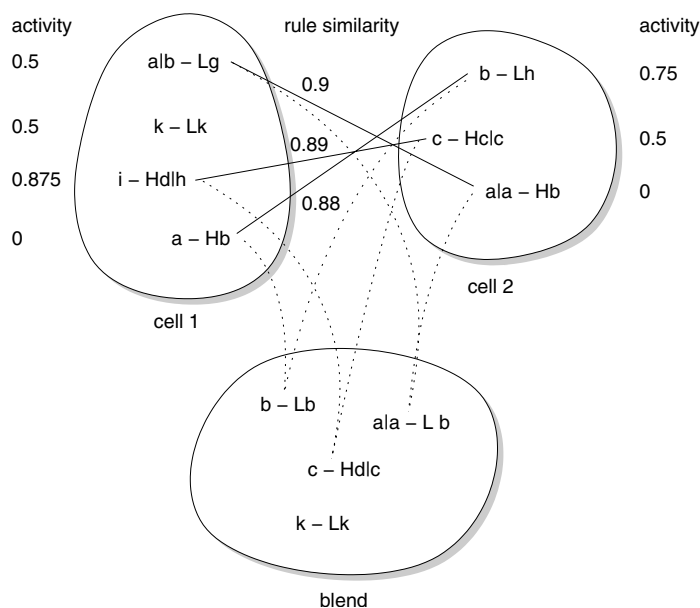


Fig. 7. Blending a new cell from two single-membrane input cells. See text for more details

established by merging the rules of the two cells together in one membrane and by letting them “float” around freely in the reactor network while they “look out” for suitable partners according to the similarity and the activity measure. Not all reactions need to be paired together with another reaction. After some time, the reaction pairs will be blended by randomly taking elements from one of the two reactions in order to form similar, yet new reactions. The resulting cell will therefore contain a mix of new reactions created on the basis of the old ones, but will also contain original elements as well. From an implementational point of view, all these steps are realized by means of special symbols in the reactions as described in Section 4.2. Note that the activity and similarity measure as well as the way the new rules are constructed depends on the application and on what information the objects and rules represent and work on. The above example should thus only be considered as a case study.

So far, we have only seen how to blend a new cell from two input cells, but how can this mechanism be used for creating good new cells that would solve a certain problem? The basic idea is similar to the principles of evolutionary algorithms (EAs). Instead of using optimality principles, each cell becomes a fitness value assigned that expresses how good a given task is solved. As with EAs, one would then maintain a population of cells where the worst cells die out and the good ones survive. Instead of applying crossover and mutation operators, we simply apply the blending mechanism to create a daughter cell from two parent cells.

In [32], we have illustrated in a simple toy application that the above described blending method works for a simple pattern classification tasks. Ex-

tending the method in order to apply it to a robot's maze navigation task, for example, would be pretty straightforward. Also, we have so far only focused on a blending mechanism for single membrane systems. Blending multi-membrane systems and testing the gradual creation of more hierarchical levels will be addressed in future work.

6 Conclusion

We have outlined an unconventional reconfigurable computer architecture enhanced with an unconventional adaptation paradigm. The resulting architecture is completely decentralized, particle based, supports the creation of hierarchical membrane systems, and represents a support for chemical blending, which on its turn can be used to create new membrane systems. In combination with a population of membrane systems and a fitness-based optimality measure, this allowed us to implement an algorithm not unlike an evolutionary algorithm in order to create membrane systems able to solve a given task.

Most of the proposed concepts have been implemented and simulated in Matlab, but so far only on their individual level (as shown in Figure 1). Simulating the entire architecture, i.e., running chemical blending on the particles, was impractical and computationally too intensive in a first step. The simulations should be considered as a proof of concept only and we are certainly far away from solving any real world problems. Clearly, many questions remain open and further research will be necessary to investigate the properties, drawbacks, and strengths of that unconventional computing architecture.

Future work will be focused in particular on further developing and investigating variants of chemical blending in order to improve the performance and to be able to solve real-world problems, such as for example a robot navigation task. A further goal is to investigate the properties of chemical reactor networks as a function of their various parameters, such as their interconnection topology, their internal storage capacity, and the parameter p_{leave} . We also plan to propose a hardware implementation of the basic particle by means of a hardware description language such as VHDL.

Acknowledgments. The author was supported by the Swiss National Science Foundation under grant PBEL2-104420.

References

1. H. Abelson, D. Allen, D. Coore, C. Hanson, E. Rauch, G. J. Sussman, and R. Weiss. Amorphous computing. *Communications of the ACM*, 43(5):74–82, May 2000.
2. E. Bilotta, D. Gross, T. Smith, T. Lenaerts, S. Bullock, H. H. Lund, J. Bird, R. Watson, P. Pantano, L. Pagliarini, H. Abbass, R. Standish, and M. A. Bedau, editors. *Alife VIII-Workshops. Workshop Proceedings of the 8th International Conference on the Simulation and Synthesis of Living Systems*. University of New South Wales, Australia, December 2002.

3. R. Brooks. The relationship between matter and life. *Nature*, 409:409–411, January 18 2001.
4. D. Coore, R. Nagpal, and R. Weiss. Paradigms for structure in an amorphous computer. Technical Report AI Memo 1614, MIT Artificial Intelligence Laboratory, October 6 1997.
5. P. Dittrich, J. Ziegler, and W. Banzhaf. Artificial chemistries—a review. *Artificial Life*, 7(3):225–275, 2001.
6. G. Fauconnier and M. Turner. Conceptual integration networks. *Cognitive Science*, 22(2):133–187, April–June 1998.
7. G. Fauconnier and M. Turner. *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books, 2002.
8. J. Goguen. An introduction to algebraic semiotics, with applications to user interface design. In Nehaniv [21], pages 242–291.
9. J. Goguen and F. Harrell. Foundations for active multimedia narrative: Semiotic spaces and structural blending. *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*, 2004. (To appear).
10. J. Goguen and F. Harrell. Style as choice of blending principles. In *Proceedings of the Symposium on Style and Meaning in Language, Art, Music and Design, 2004 AAAI Fall Symposium*, Washington DC, Oct 21–24 2004.
11. D. Gross and McMullin B. The creation of novelty in artificial chemistries. In Standish et al. [29], pages 400–408.
12. F. Gruau. Cellular encoding of genetic neural networks. Technical Report 92-21, Ecole Normale Supérieure de Lyon, Institut IMAG, 1992.
13. F. Gruau. *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. PhD thesis, Ecole Normale Supérieure de Lyon, 1994.
14. F. Gruau, Y. Lhuillier, P. Reitz, and O. Temam. BLOB computing. In S. Vassiliadis, J.-L. Gaudiot, and V. Piuri, editors, *Proceedings of the First Conference on Computing Frontiers*, pages 125–139, New York, NY, USA, 2004. ACM Press.
15. F. Gruau and P. Malbos. The Blob: A basic topological concept for hardware-free distributed computation. In C. Calude, M. J. Dinneen, and F. Peper, editors, *Unconventional Models of Computation*, volume 2509 of *Lecture Notes in Computer Science*, pages 151–163, Berlin, Heidelberg, 2002. Springer-Verlag.
16. N. J. Macias and L. J. K. Durbeck. Adaptive methods for growing electronic circuits on an imperfect synthetic matrix. *Biosystems*, 73(3):172–204, March 2004.
17. D. Mange, M. Sipper, A. Stauffer, and G. Tempesti. Toward robust integrated circuits: The embryonics approach. *Proceedings of the IEEE*, 88(4):516–540, April 2000.
18. D. Mange and M. Tomassini, editors. *Bio-Inspired Computing Machines: Towards Novel Computational Architectures*. Presses Polytechniques et Universitaires Romandes, Lausanne, Switzerland, 1998.
19. R. Nagpal. *Programmable Self-Assembly: Constructing Global Shape using Biologically-Inspired Local Interactions and Origami Mathematics*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering and Computer Science, June 2001.
20. R. Nagpal and D. Coore. An algorithm for group formation in an amorphous computer. Technical Report AI Memo 1626, MIT Artificial Intelligence Laboratory, February 16 1998.
21. C. L. Nehaniv, editor. *Computation for Metaphor, Analogy and Agents*, volume 1562 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, 1999.

22. G. Paun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000. First published in a TUCS Research Report, No 208, November 1998, <http://www.tucs.fi>.
23. G. Paun. *Membrane Computing*. Springer-Verlag, Berlin, Heidelberg, Germany, 2002.
24. G. Paun and G. Rozenberg. A guide to membrane computing. *Journal of Theoretical Computer Science*, 287(1):73–100, 2002.
25. F. C. Pereira and A. Cardoso. The horse-bird creature generation experiment. *The Interdisciplinary Journal of Artificial Intelligence and the Simulation of Behaviour*, 1(3):257–280, July 2003.
26. B. Petreska and C. Teuscher. A reconfigurable hardware membrane system. In C. Martin-Vide, G. Mauri, G. Paun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing*, volume 2933 of *Lecture Notes in Computer Science*, pages 269–285, Berlin, Heidelberg, 2004. Springer-Verlag.
27. M. C. Roco and W. S. Bainbridge, editors. *Converging Technologies for Improving Human Performance: Nanotechnology, Biotechnology, Information Technology and Cognitive Science*. World Technology Evaluation Center (WTEC), Arlington, Virginia, June 2002. NSF/DOC-sponsored report.
28. M. Sipper. *Machine Nature: The Coming Age of Bio-Inspired Computing*. McGraw-Hill, New York, 2002.
29. R. K. Standish, M. A. Bedau, and H. A. Abbass, editors. *Artificial Life VIII. Proceedings of the Eight International Conference on Artificial Life*. Complex Adaptive Systems Series. A Bradford Book, MIT Press, Cambridge, MA, 2003.
30. S. H. Strogatz. Exploring complex networks. *Nature*, 410:268–276, March 8 2001.
31. G. Tempesti, D. Roggen, E. Sanchez, Y. Thoma, R. Canham, A. Tyrrell, and J.-M. Moreno. A POetic architecture for bio-inspired hardware. In Standish et al. [29].
32. C. Teuscher. *Amorphous Membrane Blending: From Regular to Irregular Cellular Computing Machines*. PhD thesis, Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, 2004. Thesis No 2925.
33. C. Teuscher and M. Sipper. Hypercomputation: Hype or computation? *Communications of the ACM*, 45(8):23–24, August 2002.
34. S. M. Trimberger. *Field-Programmable Gate Array Technology*. Kluwer Academic Publishers, Boston, 1994.
35. A. Tyrrell, E. Sanchez, D. Floreano, G. Tempesti, D. Mange, J.-M. Moreno, J. Rosenberg, and Alessandro E. P. Villa. Poetic tissue: An integrated architecture for bio-inspired hardware. In A. M. Tyrrell, P. C. Haddow, and J. Torresen, editors, *Evolvable Systems: From Biology to Hardware. Proceedings of the 5th International Conference (ICES2003)*, volume 2606 of *Lecture Notes in Computer Science*, pages 129–140. Springer-Verlag, Berlin, Heidelberg, 2003.
36. T. Veale and D. O’Donoghue. Computation and blending. *Cognitive Linguistics*, 11(3–4):253–281, 2000.