Chemical Programming of Self-Organizing Systems

by Jean-Pierre Banâtre, Pascal Fradet and Yann Radenac

Chemical programming relies on the "chemical metaphor": data are seen as molecules and computations as chemical reactions. This programming paradigm exhibits self-organizing properties and allows the description of autonomic systems in an elegant way.

The Chemical Programming Model

This formalism was proposed to capture the intuition of computation as the global evolution of a collection of atomic values interacting freely. It can be introduced intuitively through the chemical reaction metaphor. The unique data structure is the multiset (a set possibly containing identical elements) which can be seen as a chemical solution. A simple program is made of a *reaction condition* and an *action*. Execution proceeds by replacing elements satisfying the reaction condition by the elements specified by the action. The result of such a program is obtained when a stable state is reached that is to say when no more reactions can take place. For example, the computation of the maximum element of a non empty set can be described as:

$$\textbf{replace } x, y \textbf{ by } x \textbf{ if } x \geq y$$

Any couple of elements x and y of the multiset is replaced by x if the condition is fulfilled. This process goes on till a stable state is reached, that is to say, when only the maximum element remains. Note that, in this definition, nothing is said about the order of evaluation of the comparisons. If several disjoint pairs of elements satisfy the condition, reactions can be performed in parallel.

Chemical Programming and Self-Organizing Systems

Autonomic computing provides a vision in which systems manage themselves according to some predefined goals. The essence of autonomic computing is self-organization. Like biological systems, autonomic systems maintain and adjust their operation in the face of changing components, workloads, demands, and external conditions in the face of hardware or software failures, either innocent or malicious. The autonomic system might continually monitor its own use and check for component upgrades. We believe that the chemical programming paradigm is well-suited to the descriptions of autonomic systems. It captures the intuition of a collection of cooperative components which evolve freely according to some predefined constraints (reaction rules). System self-management arises as a result of interactions between members, in the same way as ``intelligence" emerges from cooperation in colonies of biological agents.

A Self-Organizing Sorting Algorithm

Consider the general problem of a system whose state must satisfy a number of properties but which is submitted to external and uncontrolled changes. This system must constantly re-organize itself to satisfy the properties. Let us illustrate this class of problem by a simple sorting example where the system state is made of pairs (*index* : *value*) and the property of interest is that values are well-ordered (i.e. a smaller index means a smaller value). If the environment keeps adding random pairs in the state, the system must re-organize itself after each insertion of an ill-ordered element. The system is represented by a chemical solution $State = \langle sort, (i_1 : v_1), \ldots, (i_n : v_n) \rangle$, a solution made of pairs and of the following active molecule:

$$sort = \textbf{replace}\ (i : x),\ (j : y)\ \ \textbf{by}\ (i : y),\ (j : x)\ \ \textbf{if}\ i < j \text{ and } x > y$$

The molecule *sort* looks for couples of ill-ordered values and swaps them. The solution evolves up to the point where no more reactions are possible: the solution has reached a stable state and the ordering property is satisfied.

Adding new ill-ordered values breaks the ``equilibrium'' since they violate the ordering property. However, *sort* searches continuously for new ill-ordered values and makes reactions so that the state will reach a new stable state.

This very simple example shows how the Chemical Paradigm naturally expresses self-organizing systems. A program is made of a collection of rules (active molecules) which react until a stable state is reached and the corresponding invariant properties satisfied. These rules remain present and are applied (without any external intervention) as soon as the solution is unstable again.

Examples of Applications

An autonomic mail system has been described within the chemical framework. Rules ensure that all messages have reached their destination; if it is not the case some reactions occur to reach that state. The system includes rules that ensure other autonomic properties such as self-healing (rules that set and unset emergency servers in case of failures), self-optimizing (rules that balance the load between several servers), self-protecting (rules that suppress spam or viruses), self-configuration (rules that forward messages to the new address of a user), etc.

We have also specified a chemical Distributed Versioning System. In that application, several editors may edit concurrently a document made out of a set of files. The editors are distributed over a network and each one works on her/his version, makes her/his own modifications to the files and commits them locally. From time to time, two or more editors merge their modifications to propagate them. This system can be easily described with reaction rules to reflect the self-organization of the system. The system is also self-repairing: if an editor loses his local version it can revert to a previous state by synchronising with one or several other editors.

Future Plans

We are currently studying the application of this model to the coordination of program execution on grids. In a first step, applications are programmed in an abstract manner describing essentially the chemical coordination between (not necessarily chemical) software components. In a second step, chemical service programs are specifically provided to the run-time system in order to obtain the expected quality of service in terms of efficiency, reliability, security, etc.

Reference

J.-P. Banâtre, P. Fradet and Y. Radenac. 'Higher-Order Programming Style'. In Proceedings of the Workshop on Unconventional Programming Paradigms (UPP'04), LNCS 3566. Springer-Verlag.

Contact

INRIA / IRISA, Campus de Beaulieu, 35042 Rennes Cedex, France
INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 Montbonnot, France
{Jean-Pierre.Banatre, Pascal.Fradet, Yann.Radenac}@inria.fr