Fault Tolerant Autonomic Computing Systems in a Chemical Setting

Jean-Pierre Banâtre, Christine Morin, and Thierry Priol

INRIA, France
Campus de Beaulieu
35042 Rennes cedex, France
{FirstName.LastName}@inria.fr

Abstract. The chemical computing model was originally proposed as a simple and elegant parallel programming paradigm. Data are seen as molecules and computation as chemical reactions: if some molecules satisfy a predefined reaction condition, they are replaced by the product of the reaction. When no reaction is possible, a normal form is reached and the program terminates. In this paper, we describe how we can build fault tolerant autonomic systems in a chemical setting. We pay a particular attention to the chemical description of the simple and successful constructs for fault-tolerance such as the recovery block scheme by designing a generic chemical framework. Then, we apply this framework to the development of an autonomic mailing system... all this in a chemical setting.

1 Introduction

The Gamma formalism was proposed in [5] to capture the intuition of computation as the global evolution of a collection of atomic values interacting freely. Gamma can be introduced intuitively through the chemical reaction metaphor. The unique data structure in Gamma is the multiset 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 a Gamma 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 multiset can be described by the reaction rule

\texttt{replace x, y by x if } x \geq y

meaning that any couple of elements \(x\) and \(y\) of the multiset such that \(x\) is greater or equal to \(y\) is replaced by \(x\). 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, the reactions can be performed in parallel.
Gamma can be formalized as a multiset rewriting language. The literature about Gamma, as summarized in [1], is based on finite multisets of basic values. However, this basic concept can be extended by allowing elements of multisets to be reactions themselves (higher-order multisets), to have an infinite multiplicity (infinite multisets) and even to have a negative multiplicity (hybrid multisets). We describe in [4] these unconventional multiset structures (higher-order, infinite and hybrid multisets) and show how they can be interpreted in a chemical programming setting. In particular, we have presented the $\gamma$-calculus, a minimal higher-order calculus that summarizes the fundamental concepts of chemical programming. From this basic language, we have derived HOCL (the Higher Order Chemical Language), a programming language built by extending the $\gamma$-calculus with constants, operators, types and expressive patterns.

This paper describes a way of building fault tolerant autonomic systems in a chemical setting. We pay a particular attention to the chemical description of a simple and successful constructs for fault-tolerance, known as the recovery block and introduced initially by Brian Randell. We then apply the recovery block concept to the development of a simple fault tolerant autonomic mailing system entirely in a chemical setting.

2 The Higher-Order Chemical Language

The HOCL language [4] is a higher-order extension of Gamma [5] based on the $\gamma$-calculus [3]. Here, we present briefly and informally the features of HOCL used in this article. The interested reader will find a more complete and formal presentation [4]. In HOCL, programs, solutions, data and reactions are all molecules. A program is a solution of atoms $\langle A_1, \ldots, A_n \rangle$ that is, a multiset of constants, reaction rules and (sub-)solutions. The associativity and commutativity of the operator “,” formalize the Brownian motion within a chemical solution. These laws can always be used to reorganize molecules in solutions. Atoms are either basic constants (integers, booleans, etc.), pairs ($A_1: A_2$), sub-solutions ($\langle M \rangle$) or reaction rules. A reaction rule is written

$$\text{replaceone } P \text{ by } M \text{ if } C$$

where $P$ is a pattern which selects some atoms, $C$ is the reaction condition and $M$ the result of the reaction. If $P$ matches atoms which satisfy $C$, they are replaced by $M$. For example,

$$\langle \text{replaceone } x::\text{Int by } x + 1 \text{ if } x \text{ div } 2 \rangle, 4, 9, 15 \rightarrow_{\gamma} \langle 5, 9, 15 \rangle.$$

The pattern $x::\text{Int}$ matches any integer, the condition imposes the integer to be even and the action replaces it by the next odd integer. In the rest of this article, we omit types in patterns when there is no ambiguity. For example, from the