Representing Biological Systems through Multiset Rewriting

F. Martinelli[1], S. Bistarelli[1], I. Cervesato[2*], G. Lenzini[3], and R. Marangoni[4,5]

[1] Istituto di Informatica e Telematica — C.N.R.
Via G. Moruzzi 1, I-56100 Pisa — Italy
{fabio.martinelli,stefano.bistarelli}@iit.cnr.it

[2] Advanced Engineering and Sciences Division, ITT Industries, Inc.
Alexandria, VA 22303 — USA
iliano@itd.nrl.navy.mil

[3] Istituto di Scienze e Tecnologie Informatiche — C.N.R.
Via G. Moruzzi 1, I-56100 Pisa — Italy
lenzini@iei.pi.cnr.it

[4] Istituto di Biofisica — C.N.R.
Via G. Moruzzi 1, I-56100 Pisa — Italy
{roberto.marangoni}@ib.pi.cnr.it

[5] Dipartimento di Informatica, Università di Pisa,
Via F. Buonarroti 2, 56127 Pisa — Italy.

Abstract. We report on preliminary experiments at using the MultiSet Rewriting formalism MSR to specify biomolecular processes. The results are promising as MSR provides very direct and conceptually simple representations of our examples. MSR has been successfully used to describe and analyze complex distributed systems where a global states evolves as the result of local transformations, such as security protocols.

Keywords: Bio-molecular Processes, Representation, Multiset Rewriting.

1 Introduction

In the post-genomic era, the most prominent biological problems are detecting, describing and analyzing the informational flows that make a set of molecules a living organism [5]. Genomic and proteomic techniques, in fact, are producing the largest set of biological data available ever, but the problem of detecting and describing how these entities (genes and proteins) interact with each other in the complex molecular machinery of the cell has just begun being addressed. It is necessary to find easy, comprehensive, and biological-friendly *models* to describe molecules and their interactions.

It is common opinion [5] that an ideal model for biological enquiring has to satisfy three requirements:

* Cervesato was partially supported by NRL under contract N00173-00-C-2086. This work was completed while this author was visiting Princeton University.

- It must be suitable for describing metabolic networks, in order to create metabolic databases allowing the user to search for and compare biochemical pathways in living organisms (like the genomic and proteomic database are already doing).
- It must be implementable into a simulation machine, in order to realize dynamic models of metabolic pathway that allow studying possible critical situation and steady states, and generally predicting that certain conditions will happen.
- It must be possible to run dynamic simulations in which to evaluate how external agents interfere with molecules and processes, in order to infer the consequences on the metabolic network stability. This kind of applications is a useful in silico test of possible side effects of a drug.

For these reasons, proper theories and instruments of Formal Methods research community may help in defining formal models and tools (*e.g.*, see [6]), since they have been used so far to represent different kinds of relationships and dynamic interactions among objects and processes in distributes systems. In this paper we propose the use of Multiset Rewriting (MSR) [3, 2], a logic-based formalisms based on rewriting systems. MSR is offers both a formal language for a precise description of molecular interaction maps, and an execution model allowing simulation of dynamics of molecular networks with the theoretical possibility of predicting optimal values for certain parameters used in the system description. Basic mechanisms in MSR include: (a) a multiset of items, used to describe a system state, which can represent objects or resources or generic entities; (b) a set of rewriting rules which act on a state by consuming and producing items. It is our opinion that those simple and abstract mechanisms are expressive enough in describing a large class interactions happening in molecular systems.

2 Multiset Rewriting

The formal language of MultiSet Rewriting, MSR [2, 3], is given by the following grammar, defining multisets, multiset rewriting rules and rule sets:

$$\begin{array}{lrcl} \textit{Multisets} & \tilde{a}, \tilde{b}, \tilde{c}, \tilde{g} & ::= & \cdot \mid a, \tilde{a} \\ \textit{Multiset rewrite rules} & r & ::= & \tilde{g}; \tilde{a} \to \tilde{b} \\ \textit{Rule sets} & \tilde{r} & ::= & \cdot \mid r, \tilde{r} \end{array}$$

The elements of a multiset, denoted a above, are *facts* $p(\boldsymbol{t})$ where p is a predicate symbol and the terms $\boldsymbol{t} = (t_1, \ldots, t_n)$ are built from a set of symbols Σ and variables $x, y, z, \ldots$. Numerous examples will be given in the sequel. The elements in a multiset $\tilde{a}$ shall be considered unordered, but may contain replicated elements. For convenience, "$\cdot$" will be kept implicit when $\tilde{a}$ has at least one element. Similar conventions apply to rule sets.

In a rule $r = \tilde{g}; \tilde{a} \to \tilde{b}$, the multisets $\tilde{g}$, $\tilde{a}$ and $\tilde{b}$ are called the *guard*, the *antecedent* and the *consequent*, respectively. Rules with an empty guard will be simplified into $\tilde{a} \to \tilde{b}$. We will sometimes emphasize that the above rule

mentions variables $\boldsymbol{x} = (x_1, \ldots, x_n)$ by writing it $r(\boldsymbol{x}) = \tilde{g}(\boldsymbol{x}); \tilde{a}(\boldsymbol{x}) \to \tilde{b}(\boldsymbol{x})$. Then, we denote the rule obtained by substituting the variables $\boldsymbol{x}$ with terms $\boldsymbol{t} = (t_1, \ldots, t_n)$ as $r(\boldsymbol{t}) = \tilde{g}(\boldsymbol{t}); \tilde{a}(\boldsymbol{t}) \to \tilde{b}(\boldsymbol{t})$.

An MSR specification describes the situation a system is in at a certain instant as a multiset $\tilde{a}$ without any variable. This is called a *state* and written s possibly subscripted. The transformations that describe the legal evolution of the system are given as a set of rules $\tilde{r}$. We represent the fact that the system evolves from state s to state s' by using one rule r in $\tilde{r}$ as the judgment

$$\textit{Single rule application} \qquad \tilde{r} : s \longrightarrow s'$$

Operationally, this step is described by the following inference rule

$$\frac{}{(\tilde{r}, \underbrace{\tilde{g}(\boldsymbol{x}); \tilde{a}(\boldsymbol{x}) \to \tilde{b}(\boldsymbol{x})}_{r}) : \underbrace{\tilde{c}, \tilde{g}(\boldsymbol{t}), \tilde{a}(\boldsymbol{t})}_{s} \longrightarrow \underbrace{\tilde{c}, \tilde{g}(\boldsymbol{t}), \tilde{b}(\boldsymbol{t})}_{s'}}$$

In order for r to be *applicable* in state s, s must contain instances $\tilde{g}(\boldsymbol{t})$ and $\tilde{a}(\boldsymbol{t})$ of r's guard $\tilde{g}(\boldsymbol{x})$ and of its antecedent $\tilde{a}(\boldsymbol{x})$, and possibly some other facts $\tilde{c}$. If r is applicable, s' is obtained from s by removing $\tilde{a}(\boldsymbol{t})$ and replacing it with the corresponding instance of the consequent, $\tilde{b}(\boldsymbol{t})$. Notice that only the antecedent of r is altered, in particular the guard and other state components are left untouched.

Basic execution step can be chained and executed in parallel whenever the antecedents of the rules involved do not interfere. The iterated judgment is written $\tilde{r} : s \longrightarrow\!\!\!\!\!\!^{*}\ \ s'$. For some applications, the sequence of rules applied and the instantiating substitutions they use can be recorded.

Built-In Predicates Guards regulate the applicability of a rule. They are commonly used to invoke side computations that would be too tedious to model as rules. In particular, arithmetic calculations are generally performed through by invoking *built-in predicates* in guard. For example, the fact "$x + y = z$" may have the effect of calculating z as the sum of x and y, or of verifying whether $x + y$ is the number z. For clarity, we will write pseudo-predicates in quotes as just done.

3 MSR for modeling Biological Systems

Bio-chemical and molecular systems can be seen at a certain level of abstraction as a set of objects $\{O_i(\boldsymbol{s})\}$ (*e.g.*, molecules, proteins etc.), eventually with a given structure $\boldsymbol{s}$, lying in an environment where other entities $\{E_i(e)\}$ (*e.g.*, enzymes of type e) can be found. If certain environmental and parametric conditions (*e.g.*, constraints) are satisfied, a change in system may happens. Effect of this change is that certain objects are consumed (or their structure change) while other may be produced. Using MSR we are able to model many of such situations, of increasing difficulty.

Example 1. Consider the enzyme S_e (*Arginine-succinate-synthetase*) and the molecules A (*Citrulline*), P (*Aspartate*), R (*Arginine*), S (*Arginino-succinato*). The typical step of the urea cycle [7] is represented by the rule:

$$S_e, A, P \to R, S$$

Observe how this rule naturally implements the stechiometric formula usually used to describe that reactions [7]. ■

Constraints may be used to either decide new quantities in a reaction or to determine when a rule can be applied:

Example 2. Continuing to explore the urea-cycle, let us consider quantities q_1 of *Arginine*, written $R(q_1)$, q_2 of *water* $H_2O(q_2)$ and q_3 of the enzyme *Arginase* $A_e(q_3)$ which is present in a negligible quantity, that is preserved over the reaction. The final products are *Urea* U, and *Ornitine* O in quantities q_4 and q_5 respectively. One possible rule could be:

$$\text{“}\mathcal{C}_f(\tilde{q})\text{”}, \text{“}\mathcal{C}_R(\tilde{q})\text{”}, A_e(q_3); R(q_1), H_2O(q_2) \to R(q_1'), H_2O(q_2'), U(q_4), O(q_5),$$

where “$\mathcal{C}_f(\tilde{q})$” is a constraint that express the relation between the output quantities q_1', q_2', q_4 and q_5 and the input quantities q_1, q_2, q_3, while “$\mathcal{C}_R(\tilde{q})$” is a built-in predicate used to determine when the rule may be applied. For example “$\mathcal{C}_R(\tilde{q})$” may express that it is required $q_1 \ll q_2$ in order the reaction happens. Notice that $A_e(q_3)$ is written in the guard of this rule since it doesn't change during application. ■

Structured terms in the predicates may be used to reflect the structure of the domain. For example:

Example 3. Consider a predicate $M(c)$, where M stands for molecule and $c = \langle p, c_1 \rangle$ is a term representing a chain of proteins, e.g., p is the “first” protein and c_1 the rest of the chain. Consider also a predicate $E(e)$ which stands for an enzyme of type e. Assume the enzyme $E(\texttt{cut})$ is able to cut one piece of the chain. Then, a MSR rule that models this behavior may be:

$$E(\texttt{cut}); M(\langle p, c_1 \rangle) \to M(c_1), M(p)$$

By using this rule several time, for example is possible, starting from the initial state

$$E(\texttt{cut}), M(\langle \texttt{pro}_1, \langle \texttt{pro}_2, \texttt{pro}_3 \rangle \rangle)$$

to reach a state

$$E(\texttt{cut}), M(\texttt{pro}_1), M(\texttt{pro}_2), M(\texttt{pro}_3).$$

If another enzyme $E(bin)$ attaches a new protein to an existing chain, we may also have the rule

$$E(\texttt{bin}); M(c_1), M(p) \to M(\langle p, c_1 \rangle)$$

■

4 Conclusions and future work

The language MSR has been used to study various forms of concurrent distributed computation [2, 1]. We propose MSR as a natural framework to model bio-chemical reactions. It has a clear and rigorous semantics. Moreover, its rules are readily understandable and seem quite close to textual descriptions of chemical reactions, at a functional level. So far, our experience on translating informal graphical representations of metabolic paths into MSR has been positive.

From the analysis and simulation point of view, we plan to explore a line of research already used with good success for security protocols analysis. Indeed, many analysis tools compile high level specifications into simpler MSR specifications [4]. Due to the simplicity and uniformity of the MSR rewrite mechanism, many researchers feel easier to develop analysis algorithms for MSR. In particular, we are currently modeling a stochastic version of MSR; moreover, we plan to study an encoding from process algebras akin to [6] into stochastic MSR (similarly to the encoding we proposed in [1] for security analysis).

References

[1] S. Bistarelli, I. Cervesato, G. Lenzini, and F. Martinelli. Relating process algebras and multiset rewriting (for example for security protocol analysis). Technical report, Istituto di Scienza e Tecnologie dell'Informazione (ISTI-CNR), 2002. To appear.

[2] I. Cervesato. A Specification Language for Crypto-Protocols based on Multiset Rewriting, Dependent Types and Subsorting. In G. Delzanno, S. Etalle, and M. Gabbrielli, editors, *Workshop on Specification, Analysis and Validation for Emerging Technologies — SAVE'01*, pages 1–22, Paphos, Cyprus, 2001.

[3] Iliano Cervesato, Nancy Durgin, Patrick D. Lincoln, John C. Mitchell, and Andre Scedrov. A Meta-Notation for Protocol Analysis. In *12th Computer Security Foundations Workshop — CSFW-12*, pages 55–69, Mordano, Italy, 28–30 June 1999. IEEE Computer Society Press.

[4] Grit Denker and Jonathan K. Millen. CAPSL Intermediate Language. In N. Heintze and E. Clarke, editors, *Proceedings of the Workshop on Formal Methods and Security Protocols — FMSP*, Trento, Italy, July 1999.

[5] A. Pevzner. *Computational Molecular Biology.* MIT press, 2000.

[6] C. Priami, A. Regev, E. Shapiro, and W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80, 2001.

[7] L. Stryer. *Biochemistry.* W. H. Freeman and Company, New York, 1981.