

# An Intermediate Language for the Simulation of Biological Systems

Giulio Caravagna

MSc Thesis

12th, October 2007

# Outline of the talk

## Introduction

Multiset Rewriting (MSR)

## String Multiset Rewriting (SMSR)

Terms, Patterns and Rules

Semantics

## SMSR as an Intermediate Language

Intuitions

Encoding CLS+

## Conclusions

# Multiset Rewriting (MSR)

$$M \equiv a \mid a \mid a \mid b \mid b \mid c$$

$$R : a \mid b \mapsto c$$

$$M' \equiv a \mid a \mid b \mid c \mid c$$

- ▶ syntax similar to that of process algebras
- ▶ stochastic semantics can be easily formalized
- ▶ chemical reactions can be described by using MSR

# High-level languages

- ▶  $\pi$ -calculus,  $\kappa$ -calculus, P-Systems, Brane Calculi, Bio Ambients, CLS, Beta Binders, ...
- ▶ describe biological systems at different levels of abstraction
  - ▶ membranes, sequences, ...
- ▶ their semantics are transition systems where the states are terms
- ▶ adding and implementing stochastic behavior may not be trivial
- ▶ developing ad-hoc simulators

# The idea

*"Engineering and implementing ad-hoc simulators for higher-level languages"*

against

*"Defining an intermediate language, implementing its semantics and encoding higher-level languages"*

Is the latter always possible? There exist restrictions? Is the encoding always less expensive? Finally, encodings must be proved correct . . .

# SMSR: Main features

- ▶ SMSR as a natural extension of MSR
  - ▶ objects are strings built over an alphabet
  - ▶ state of the simulation is a multiset of strings
  - ▶ rewriting rules may contain variables
    - ▶ element, sequence and multiset variables
    - ▶ in the process of instantiating *free* variables semantics provide generation of *fresh* data
  - ▶ MSR enriched with a maximal matching operator
    - ▶ useful with a general encoding technique

# SMSR: Syntax and Rewrite Rules

We assume an infinite alphabet  $\mathcal{E}$ . **Multisets**  $M$  and **Strings**  $S$  of SMSR are given by the following grammar:

$$\begin{array}{l} M ::= S \mid M \mid M \\ S ::= \epsilon \mid e \mid S \cdot S \end{array}$$

where  $e$  is a generic element of  $\mathcal{E}$ , and  $\epsilon$  is the empty string.

The operators are:

$S \cdot S$  : String concatenation

$M \mid M$  : Multiset union (juxtaposition)

The structural congruence relation  $\equiv$  on terms expresses the associativity of  $\cdot$  and  $\mid$ , the commutativity of the latter, and the neutral role of  $\epsilon$ .

Let us consider variables of three kinds:

- ▶ sequence variables  $(\tilde{x}, \tilde{y}, \tilde{z}, \dots)$
- ▶ element variables  $(x, y, z, \dots)$
- ▶ multiset variables  $(X, Y, Z, \dots)$

**Multiset Patterns**  $MP$  and **String Patterns**  $SP$  of SMSR are given by the following grammar:

$$\begin{aligned} MP & ::= SP \mid MP \mid MP \mid \{SP\}_X \mid \{SP\}_{SP} \\ SP & ::= \epsilon \mid e \mid SP \cdot SP \mid \tilde{x} \mid x \end{aligned}$$

where  $e$  is a generic element of  $\mathcal{E}$ .

A **Rewrite Rule** is a pair  $(MP, MP')$ , denoted  $MP \mapsto MP'$ , where  $MP, MP'$  are multiset patterns and  $MP \neq \epsilon$ .

# The maximal matching operator

$$M \equiv a \cdot b \mid a \cdot b \cdot a \mid a \cdot c \cdot b \mid a \cdot c \cdot b$$

- ▶  $\{a \cdot b\}_X$  represents the multiset  $a \cdot b \mid a \cdot b \cdot a$
- ▶  $\{a \cdot b\}_a$  represents the singleton  $a \cdot b \cdot a$
- ▶  $\{a \cdot c\}_b$  represents the singleton  $a \cdot c \cdot b$  (*incorrect*)
  - ▶ we should count the occurrences using  $\{a \cdot c\}_b \mid \{a \cdot c\}_b$

# The maximal matching operator

$$M \equiv a \cdot b \mid a \cdot b \cdot a \mid a \cdot c \cdot b \mid a \cdot c \cdot b$$

- ▶  $\{a \cdot b\}_X$  represents the multiset  $a \cdot b \mid a \cdot b \cdot a$
- ▶  $\{a \cdot b\}_a$  represents the singleton  $a \cdot b \cdot a$
- ▶  $\{a \cdot c\}_b$  represents the singleton  $a \cdot c \cdot b$  (*incorrect*)
  - ▶ we should count the occurrences using  $\{a \cdot c\}_b \mid \{a \cdot c\}_b$

We define some auxiliary functions.

**Pattern Expansion Function**  $\langle - \rangle_\rho$  is recursively defined as follows:

$$\langle SP \rangle_\rho = SP$$

$$\langle \{ \{ SP \}_X \} \rangle_\rho = SP \cdot \tilde{x}_1 \mid \dots \mid SP \cdot \tilde{x}_{\rho(X)} \quad \text{where } \tilde{x} \in \mathcal{V}_S$$

$$\langle \{ \{ SP_1 \}_{SP_2} \} \rangle_\rho = SP_1 \cdot SP_2 \quad \text{where } SP_2 \in \mathcal{SP}$$

$$\langle MP_1 \mid MP_2 \rangle_\rho = \langle MP_1 \rangle_\rho \mid \langle MP_2 \rangle_\rho$$

- ▶  $Var(M)$  the set of variables that appear in  $M$ ;
- ▶  $Symbols(M)$  the set of elements of  $\mathcal{E}$  that appear in  $M$ ;
- ▶  $FV(M \mapsto M') = \{v \mid v \in Var(M') \wedge v \notin Var(M)\}$ ;
- ▶  $BV(M \mapsto M') = Var(R) \setminus FV((M, M')) = Var(M)$ .

# SMSR: Formal Semantics

Given a finite set of rewrite rules  $\mathcal{R}$  the semantics of SMSR is the least labeled transition relation  $\xrightarrow{\xi, \zeta}$  closed with respect to  $\equiv$  and satisfying the following inference rules:

$$(1) \frac{\begin{array}{l} R : M_1 \mapsto M_2 \in \mathcal{R} \quad \sigma \in \Sigma \\ \exists \rho. (\langle M_1 \rangle_\rho)\sigma \equiv M \wedge (\langle M_2 \rangle_\rho)\sigma \equiv M' \\ (\text{Symbols}(\sigma(BV(R))) \cup \text{Symbols}(R)) \cap \text{Symbols}(\sigma(FV(R))) = \emptyset \end{array}}{M \xrightarrow{\{SP\sigma \mid \{SP\}_\sigma \in M'\}, \text{Symbols}(\sigma(FV(R)))} M'}$$

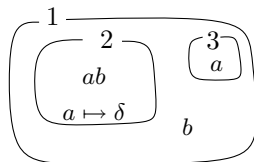
$$(2) \frac{\begin{array}{l} M \xrightarrow{\xi, \zeta} M' \quad \zeta \cap \text{Symbols}(M'') = \emptyset \\ \forall S \in \xi. \nexists S' \in \mathcal{S}. (S \cdot S') \in M'' \end{array}}{M'' \mid M \xrightarrow{\xi, \zeta} M'' \mid M'}$$

# A general encoding technique

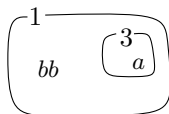
- ▶ term-based languages
  - ▶ a term is a tree (abstract syntax tree)
- ▶ we change point of view
  - ▶ a tree is a multiset of paths from the root to the leaves
  - ▶ we define labels for syntactic operators
  - ▶ the tree becomes a multiset

# Encoding: An Example

P-System

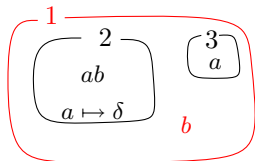


Result of applying the rule



# Encoding: An Example

P-System

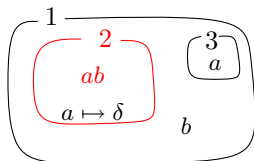


Its encoding into SMSR

$1 \cdot b$

# Encoding: An Example

P-System

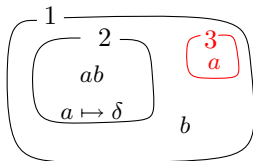


Its encoding into SMSR

$$1 \cdot b | 1 \cdot 2 \cdot a | 1 \cdot 2 \cdot b$$

# Encoding: An Example

P-System

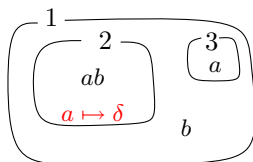


Its encoding into SMSR

$1 \cdot b | 1 \cdot 2 \cdot a | 1 \cdot 2 \cdot b | 1 \cdot 3 \cdot a$

# Encoding: An Example

P-System

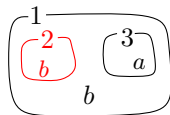


Its encoding into SMSR

$1 \cdot b | 1 \cdot 3 \cdot a | 1 \cdot 2 \cdot a | 1 \cdot 2 \cdot b$

$1 \cdot 2 \cdot a \mapsto \epsilon$

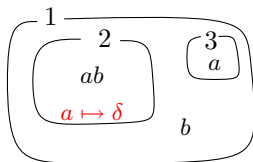
Result of applying the rule (**WRONG**)



$1 \cdot b | 1 \cdot 3 \cdot a | 1 \cdot 2 \cdot b$

# Encoding: An Example

P-System



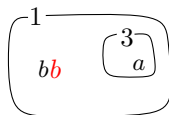
Its encoding in SMSR

$1 \cdot 2 \cdot a | 1 \cdot 2 \cdot b | 1 \cdot 3 \cdot a | 1 \cdot b$

~~$1 \cdot 2 \cdot a \mapsto \epsilon$~~

$1 \cdot 2 \cdot a | \{1 \cdot 2\}_X \mapsto \{1\}_X$

Result of applying the rule



$1 \cdot b | 1 \cdot b | 1 \cdot 3 \cdot a$

# CLS+: Features

- ▶ a term-rewriting language
  - ▶ easy and expressive syntax (terms, patterns, variables,..)
  - ▶ stochastic semantic is not easy (SCLS)
- ▶ possibility of defining membranes
  - ▶ membranes contain multisets of sequences
  - ▶ membranes can be nested
- ▶ an ad-hoc simulator has been developed
  - ▶ what about the performances?

# The Syntax of CLS+

**Terms**  $T$ , **Branes**  $B$  and **Sequences**  $S$  of CLS+ are given by the following grammar:

$$\begin{aligned} T &::= S \mid (B)^L \mid T \mid T \\ B &::= S \mid B \mid B \\ S &::= \epsilon \mid a \mid S \cdot S \end{aligned}$$

where  $a$  is a generic element of a alphabet. The operators are:

- $S \cdot S$  : Sequencing
- $(B)^L$  : Looping
- $T_1 \mid T_2$  : Containment ( $T_1$  contains  $T_2$ )
- $T \mid T$  : Parallel composition (juxtaposition)

- ▶ **Patterns** are terms enriched with variables
- ▶ **Rules** are pairs of patterns

# CLS+: Encoding

- ▶ there exist restrictions
- ▶ only the containment operator encoded
  - ▶ labels used into encoding are enriched with identifiers
  - ▶ possibility of generating fresh data is used to generate fresh identifiers
- ▶ in the encoding of term variables we use the maximal matching operator

## Encoding Patterns: use of $\{SP_1\}_{SP_2}$

Given  $T \equiv (a \mid b)^L \rfloor \epsilon$  its encoding is

$$\lfloor T \rfloor = \overline{\lambda_1} \cdot a \mid \overline{\lambda_1} \cdot b$$

a CLS+ rule  $R : (a)^L \mapsto \epsilon$  cannot be applied to  $T$ .

- ▶ (*incorrect*) encode  $R$  with  $\overline{\lambda_x} \cdot a \mapsto \epsilon$ ;  $\lfloor T \rfloor$  is rewritten in  $\overline{\lambda_1} \cdot b$  which represents the CLS+ term  $(b)^L \rfloor \epsilon$ .
- ▶ (*correct*) encode  $R$  with  $\{\overline{\lambda_x}\}_a \mapsto \epsilon$ ;  $\lfloor T \rfloor$  cannot be rewritten.

## Encoding Patterns: use of $\{\overline{SP_1}\}_{SP_2}$

Given  $T \equiv (a \mid b)^L \rfloor \epsilon$  its encoding is

$$\lfloor T \rfloor = \overline{\lambda_1} \cdot a \mid \overline{\lambda_1} \cdot b$$

a CLS+ rule  $R : (a)^L \mapsto \epsilon$  cannot be applied to  $T$ .

- ▶ (*incorrect*) encode  $R$  with  $\overline{\lambda_x} \cdot a \mapsto \epsilon$ ;  $\lfloor T \rfloor$  is rewritten in  $\overline{\lambda_1} \cdot b$  which represents the CLS+ term  $(b)^L \rfloor \epsilon$ .
- ▶ (*correct*) encode  $R$  with  $\{\overline{\lambda_x}\}_a \mapsto \epsilon$ ;  $\lfloor T \rfloor$  cannot be rewritten.

## Encoding Patterns: use of $\{SP_1\}_{SP_2}$

Given  $T \equiv (a \mid b)^L \rfloor \epsilon$  its encoding is

$$\lfloor T \rfloor = \overline{\lambda_1} \cdot a \mid \overline{\lambda_1} \cdot b$$

a CLS+ rule  $R : (a)^L \mapsto \epsilon$  cannot be applied to  $T$ .

- ▶ (*incorrect*) encode  $R$  with  $\overline{\lambda_x} \cdot a \mapsto \epsilon$ ;  $\lfloor T \rfloor$  is rewritten in  $\overline{\lambda_1} \cdot b$  which represents the CLS+ term  $(b)^L \rfloor \epsilon$ .
- ▶ (*correct*) encode  $R$  with  $\{\overline{\lambda_x}\}_a \mapsto \epsilon$ ;  $\lfloor T \rfloor$  cannot be rewritten.

# Proof of equivalence

The following theorem states correctness and completeness of the encoding of CLS+ in SMSR with respect to their semantics.

## Theorem

*For any CLS+ terms  $T$  and  $T'$*

$$T \rightarrow T' \Leftrightarrow \exists \xi, \zeta. [T] \xrightarrow{\xi, \zeta} [T']$$

# Conclusions

- ▶ SMSR as an intermediate language
  - ▶ SMSR as an extension of MSR
  - ▶ strings as multisets elements
  - ▶ generation of fresh data
  - ▶ maximal matching operator
  
- ▶ Higher-level formalisms can be translated in SMSR
  - ▶ we have shown encoding of CLS+ in SMSR
  - ▶ (future work) encoding other languages (i.e. P-Systems,...)
  - ▶ (future work) developing an ad-hoc simulator for SMSR

## APPENDIX

## The Semantics of CLS+

The semantics of CLS+ ensures that on membrane we may have multisets of objects which cannot be membranes themselves

Given a set of rewrite rules  $\mathcal{R} \subseteq \mathfrak{R}$  the *semantics* of CLS is given by the following rules

$$\frac{(P_1, P_2) \in \mathcal{R} \quad P_1\sigma \neq \epsilon \quad \sigma \in \Sigma}{P_1\sigma \rightarrow P_2\sigma}$$

$$\frac{T_1 \rightarrow T_2}{T \mid T_1 \rightarrow T \mid T_2}$$

$$\frac{T_1 \rightarrow T_2}{(B)^L \mid T_1 \rightarrow (B)^L \mid T_2}$$

$$\frac{(BP_1, BP_2) \in \mathcal{R}_B \quad BP_1\sigma \neq \epsilon \quad \sigma \in \Sigma}{BP_1\sigma \rightarrow_B BP_2\sigma}$$

$$\frac{B_1 \rightarrow_B B_2}{B \mid B_1 \rightarrow_B B \mid B_2}$$

$$\frac{B_1 \rightarrow_B B_2}{(B_1)^L \mid T \rightarrow (B_2)^L \mid T}$$

## CLS+: An Example

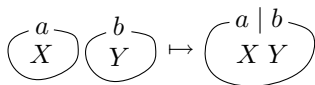
$$T \equiv ((a)^L \rfloor (a)) \mid ((b)^L \rfloor (b))$$



$$R_1 : a \mid b \mapsto c$$

$$a \ b \mapsto \ c$$

$$R_2 : ((a)^L \rfloor (X)) \mid ((b)^L \rfloor (Y)) \mapsto (a \mid b)^L \rfloor (X \mid Y)$$



$$T \equiv ((a)^L \rfloor (a)) \mid ((b)^L \rfloor (b))$$

$$\rightarrow (a \mid b)^L \rfloor (a \mid b) \quad \text{applying rule } (R_2)$$

$$\rightarrow (a \mid b)^L \rfloor (c) \quad \text{applying rule } (R_1)$$

$$\rightarrow (c)^L \rfloor (c) \quad \text{applying rule } (R_1)$$

## Encoding Terms

The encoding of CLS+ terms into multisets of strings is given by the function  $\lfloor \_ \rfloor : \mathcal{T} \rightarrow \mathcal{M} \times \wp(\mathbb{N})$  recursively defined as follows:

$$\lfloor S \rfloor = (S, \emptyset)$$

$$\begin{aligned} \lfloor T_1 \mid T_2 \rfloor &= (M_1 \mid M_2, I_1 \cup I_2) \\ &\text{where } \lfloor T_i \rfloor = (M_i, I_i) \text{ and } I_1 \cap I_2 = \emptyset \end{aligned}$$

$$\begin{aligned} \lfloor (B)^L \rfloor T \rfloor &= (\overline{\lambda}_i \triangleright M_1 \mid \lambda_i \triangleright M_2, I_1 \cup I_2) \\ &\text{where } \lfloor B \rfloor = (M_1, I_1), \lfloor T \rfloor = (M_2, I_2) \\ &\text{and } i \in \mathbb{N} \setminus (I_1 \cup I_2) \end{aligned}$$

$$\lfloor (a \cdot b)^L \rfloor (a \mid (b \cdot c)^L \rfloor (b)) \rfloor = \overline{\lambda}_1 \cdot a \cdot b \mid \lambda_1 \cdot a \mid \lambda_1 \cdot \overline{\lambda}_2 \cdot b \cdot c \mid \lambda_1 \cdot \lambda_2 \cdot b$$

## Encoding Patterns 1/2

The encoding of CLS+ patterns into multiset patterns is given by the function  $\llbracket \_ \rrbracket : \mathcal{P} \rightarrow \mathcal{MP} \times \mathcal{V}_{\mathcal{E}} \times \mathcal{V}_{\mathcal{E}}$  recursively defined as follows:

$$\llbracket SP \rrbracket = (SP, \emptyset, \text{Var}(SP))$$

$$\llbracket v \rrbracket = (\{\epsilon\}_v, \emptyset, \{v_i \mid i \in \mathbb{N}\}) \quad \forall v \in TV \cup BV$$

$$\begin{aligned} \llbracket P_1 \mid P_2 \rrbracket &= (MP_1 \mid MP_2, \Sigma_1 \cup \Sigma_2, \Sigma'_1 \cup \Sigma'_2) \\ &\text{where } \llbracket P_i \rrbracket = (MP_i, \Sigma_i, \Sigma'_i) \text{ and } \Sigma_1 \cap \Sigma_2 = \emptyset \end{aligned}$$

$$\begin{aligned} \llbracket (BP)^L \rrbracket \mid P \rrbracket &= (\overline{\lambda_x} \triangleright MP_1 \mid \lambda_x \triangleright MP_2, \{x\} \cup \Sigma_2, \Sigma'_1 \cup \Sigma'_2) \\ &\text{where } \llbracket BP \rrbracket = (MP_1, \emptyset, \Sigma'_1), \llbracket P \rrbracket = (MP_2, \Sigma_2, \Sigma'_2) \\ &\text{and } x \in \mathcal{V}_{\mathcal{E}} \setminus (\Sigma_1 \cup \Sigma'_1 \cup \Sigma_2 \cup \Sigma'_2) \end{aligned}$$

The signature contains variables and we use an auxiliary function  $\llbracket \_ \rrbracket$ .

## Encoding Patterns 2/2

The auxiliary encoding function  $(\lfloor \_ ) : \mathcal{P} \rightarrow \mathcal{MP} \times \mathcal{V}_E \times \mathcal{V}_E$  is recursively defined as follows:

$$(\lfloor SP \rfloor) = (\{\{\epsilon\}\}_{SP}, \emptyset, \text{Var}(SP))$$

$$(\lfloor v \rfloor) = (\{\{\epsilon\}\}_v, \emptyset, \{v_i \mid i \in \mathbb{N}\}) \quad \forall v \in TV \cup BV$$

$$\begin{aligned} (\lfloor P_1 \mid P_2 \rfloor) &= (MP_1 \mid MP_2, \Sigma_1 \cup \Sigma_2, \Sigma'_1 \cup \Sigma'_2) \\ &\text{where } (\lfloor P_i \rfloor) = (MP_i, \Sigma_i, \Sigma'_i) \text{ and } \Sigma_1 \cap \Sigma_2 = \emptyset \end{aligned}$$

$$\begin{aligned} (\lfloor (BP)^L \rfloor P \rfloor) &= (\overline{\lambda_x} \triangleright MP_1 \mid \lambda_x \triangleright MP_2, \{x\} \cup \Sigma_2, \Sigma'_1 \cup \Sigma'_2) \\ &\text{where } (\lfloor BP \rfloor) = (MP_1, \emptyset, \Sigma'_1), (\lfloor P \rfloor) = (MP_2, \Sigma_2, \Sigma'_2) \\ &\text{and } x \in \mathcal{V}_E \setminus (\Sigma_1 \cup \Sigma'_1 \cup \Sigma_2 \cup \Sigma'_2) \end{aligned}$$

We encode sequences inside  $(\_)^L$  operator using the second form of the maximal matching operator.

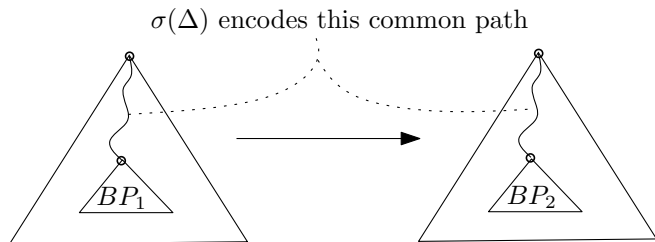
## Encoding Brane Rules

Brane rules  $BP_1 \mapsto BP_2 \in \mathfrak{R}_B$  can be applied everywhere in a CLS+ term.

Their encoding is

$$\Delta \triangleright MP_1 \mapsto \Delta \triangleright MP_2$$

where  $\llbracket BP_1 \rrbracket = (MP_1, \Sigma_1, \Sigma'_1)$ ,  $\llbracket BP_2 \rrbracket = (MP_2, \Sigma_2, \Sigma'_2)$ ,  
 $\Sigma_1 \cap \Sigma_2 = \emptyset$  and  $\Delta \notin \Sigma_1 \cup \Sigma'_1 \cup \Sigma_2 \cup \Sigma'_2$ .



## Encoding Non-brane Rules

Non-brane rules  $P_1 \mapsto P_2$  can be applied only either to the components of a parallel composition at the top level of the term, or to the components of a parallel composition inside some looping sequence.

Their encoding is

$$MP_1 \mapsto MP_2 \quad \text{and} \quad \Delta \cdot \lambda_x \triangleright MP_1 \mapsto \Delta \cdot \lambda_x \triangleright MP_2$$

where  $\llbracket P_1 \rrbracket = (MP_1, \Sigma_1, \Sigma'_1)$ ,  $\llbracket P_2 \rrbracket = (MP_2, \Sigma_2, \Sigma'_2)$ ,  $\Sigma_1 \cap \Sigma_2 = \emptyset$  and  $\{\Delta, x\} \cap (\Sigma_1 \cup \Sigma'_1 \cup \Sigma_2 \cup \Sigma'_2) = \emptyset$ .

## A Simple Example

Given  $T$ ,  $R_1$  and  $R_2$  their encoding is

$$M \equiv \bar{\lambda}_1 \cdot a \mid \lambda_1 \cdot a \mid \bar{\lambda}_2 \cdot b \mid \lambda_2 \cdot b$$

$$(1) \quad \Delta \cdot a \mid \Delta \cdot b \mapsto \Delta \cdot c$$

$$(2) \quad \{\{\Delta \cdot \lambda_w \cdot \bar{\lambda}_x\}_a \mid \{\Delta \cdot \lambda_w \cdot \lambda_x\}_X \mid \{\Delta \cdot \lambda_w \cdot \bar{\lambda}_y\}_b \mid \{\Delta \cdot \lambda_w \cdot \lambda_y\}_Y \\ \mapsto \{\{\Delta \cdot \lambda_w \cdot \bar{\lambda}_z\}_a \mid \{\Delta \cdot \lambda_w \cdot \bar{\lambda}_z\}_b \mid \{\Delta \cdot \lambda_w \cdot \lambda_z\}_X \mid \{\Delta \cdot \lambda_w \cdot \lambda_z\}_Y$$

$$(3) \quad \{\{\bar{\lambda}_x\}_a \mid \{\lambda_x\}_X \mid \{\bar{\lambda}_y\}_b \mid \{\lambda_y\}_Y \mapsto \{\{\bar{\lambda}_z\}_a \mid \{\bar{\lambda}_z\}_b \mid \{\lambda_z\}_X \mid \{\lambda_z\}_Y$$

A SMSR computation corresponding to the shown CLS+ computation is

$$T \equiv \bar{\lambda}_1 \cdot a \mid \lambda_1 \cdot a \mid \bar{\lambda}_2 \cdot b \mid \lambda_2 \cdot b$$

$$\rightarrow \bar{\lambda}_3 \cdot a \mid \bar{\lambda}_3 \cdot b \mid \lambda_3 \cdot a \mid \lambda_3 \cdot b \quad \text{applying rule (3)}$$

$$\rightarrow \bar{\lambda}_3 \cdot a \mid \bar{\lambda}_3 \cdot b \mid \lambda_3 \cdot c \quad \text{applying rule (1)}$$

$$\rightarrow \bar{\lambda}_3 \cdot c \mid \lambda_3 \cdot c \quad \text{applying rule (1)}$$

## References

- ▶ G. Caravagna.  
*"An Intermediate Language for the Simulation of Biological Systems"*  
MSC Thesis, Department of Computer Science, University of Pisa, 2007
  
- ▶ R. Barbuti, G. Caravagna, A. Maggiolo–Schettini and P. Milazzo.  
*"An Intermediate Language for the Simulation of Biological Systems"*  
1th Int. Workshop From Biology to Concurrency and Back (FBTC'07), ENTCS, to appear.