Carlos Martín-Vide
Giancarlo Mauri
Gheorghe Păun
Grzegorz Rozenberg
Arto Salomaa (Eds.)

LNCS 2933

# Membrane Computing

International Workshop, WMC 2003
Tarragona, Spain, July 2003
Revised Papers

Springer

Carlos Martín-Vide Giancarlo Mauri
Gheorghe Păun Grzegorz Rozenberg
Arto Salomaa (Eds.)

# Membrane Computing

International Workshop, WMC 2003
Tarragona, Spain, July 17-22, 2003
Revised Papers



Springer

Volume Editors

Carlos Martín-Vide
Rovira i Virgili University
Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
E-mail: cmv@astor.urv.es

Giancarlo Mauri
Università degli Studi di Milano-Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione
Via Bicocca degli Arcimboldi 8, 20136 Milano, Italy
E-mail: mauri@disco.unimib.it

Gheorghe Păun
Institute of Mathematics of the Romanian Academy
P.O. Box 1-764, 70700 Bucureşti, Romania
and
Rovira i Virgili University, Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain
E-mail: gp@astor.urv.es

Grzegorz Rozenberg
Leiden University, Leiden Center of Advanced Computer Science (LIACS)
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
E-mail: rozenber@liacs.nl

Arto Salomaa
Turku Centre for Computer Science, TUCS
Leminkäisenkatu 14, 20520 Turku, Finland
E-mail: asalomaa@cs.utu.fi

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress.

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data is available in the Internet at <http://dnb.ddb.de>.

CR Subject Classification (1998): F.1, F.4, I.6, J.3

ISSN 0302-9743
ISBN 3-540-20895-X Springer-Verlag Berlin Heidelberg New York



Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 10981757 06/3142 5 4 3 2 1 0

Lecture Notes in Computer Science 2933
Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

Springer
*Berlin*
*Heidelberg*
*New York*
*Hong Kong*
*London*
*Milan*
*Paris*
*Tokyo*

# Preface

This volume is based on papers presented at the **Workshop on Membrane Computing, WMC 2003**, which took place in Tarragona, Spain, in the period July 17–July 22, 2003. This was the Fourth Annual Membrane Computing Workshop, and the first one held outside Romania. The first three meetings were organized in Curtea de Argeş, Romania – they took place in August 2000 (with the proceedings published in *Lecture Notes in Computer Science*, Vol. 2235), in August 2001 (with a selection of papers published as a special issue of *Fundamenta Informaticae*, Vol. 49, Nos. 1–3, 2002), and in August 2002 (with the proceedings published in *Lecture Notes in Computer Science*, Vol. 2597).

The 2003 workshop was the second workshop of the Molecular Computing Network (MolCoNet) funded by the EU Commission in the Fifth Framework Program Information Society Technologies (project number IST–2001–32008). The preproceedings of WMC 2003 were published as Technical Report 28/03 of the Research Group on Mathematical Linguistics from Rovira i Virgili University, Tarragona, and they were available during the workshop.

The current volume contains only a selection of the papers from the preproceedings. Moreover, the selected papers have been significantly modified/improved according to the really vivid discussions that took place during the workshop — all the selected papers were additionally refereed. The papers in the volume cover all the main directions of research in membrane computing, ranging from topics in mathematics and theoretical computer science, to (potential) applications in biology, sorting, ranking, linguistics, and computer graphics. Several implementations/simulations on computers, computer networks, or electronic reconfigurable hardware are also presented. Thus, the volume is a faithful illustration of the current state of research in membrane computing (a good source of information about membrane computing is the Web page `http://psystems.disco.unimib.it`).

The workshop was organized by the Research Group on Mathematical Linguistics from Rovira i Virgili University, Tarragona, Spain, under the auspices of the European Molecular Computing Consortium (EMCC). The program committee consisted of Carlos Martín-Vide (Tarragona, Spain), Giancarlo Mauri (Milan, Italy), Gheorghe Păun (Bucharest, Romania, and Tarragona, Spain), Grzegorz Rozenberg (Leiden, The Netherlands, and Boulder, Colorado, USA), and Arto Salomaa (Turku, Finland).

The editors are indebted to the contributors and to Springer-Verlag for the efficient cooperation in the timely production of this volume.

The workshop received financial support from a number of sources: MolCoNet Project IST-2001-32008 funded by the European Union, Project TIC2002-04220-C03-02 funded by the Spanish Ministry of Science and Technology, and the Research Group on Mathematical Linguistics of Rovira i Virgili University.

November 2003

Carlos Martín-Vide
Giancarlo Mauri
Gheorghe Păun
Grzegorz Rozenberg
Arto Salomaa

# Table of Contents

# Proton Pumping P Systems

Artiom Alhazov[1,2*] and Matteo Cavaliere[1**]

[1] Research Group on Mathematical Linguistics
Rovira i Virgili University
Pl. Imperial Tarraco 1, 43005 Tarragona, Spain
{artiome.alhazov, matteo.cavaliere}@estudiants.urv.es
[2] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Str. Academiei 5, Chişinău, MD 2028, Moldova
artiom@math.md

**Abstract.** We propose here a (biologically inspired) model of P system called *proton pumping P system* that is a special case of evolution–communication P system. In cell biology there are transport mechanisms, involving protons. We generalize this idea by considering a few different types of protons. A proton pumping P system is, essentially, an evolution–communication P system where a special subset of symbol-objects (called protons) is used. In such a system we have simple evolution rules (classical evolution rules without target indications), symport and antiport rules that exchange some objects (among them, possibly, other protons) for a proton; taking inspiration from biology, this particular type of antiports is often called *proton pumping rules.*

We show that, as expected, the new model is universal, using non-cooperative rules, symport and antiport rules of weight one, and *enough* types of protons available for the computation. If we decrease the number of types of protons to one or two, then the model is at least as powerful as ET0L system, provided that (total) *weak* or *strong* priority of antiport rules over symport and evolution rules are used.

Finally, we consider some *descriptional complexity* measures (again, inspired from biology) for the newly introduced model.

## 1 Introduction

In this paper we investigate proton pumping P systems. They are evolution–communication P systems, [3], with some restrictions inspired by the biology (in what follows we refer to [5] for the elements of membrane computing, to [1] and [6] for the elements related to cellular biology).

* This author's work was supported by the research grant 2001CAJAL-BURV4 from Rovira i Virgili University.

** This author's work was supported by the Spanish Ministry of Culture, Education and Sport under the Programa Nacional de Formación de Profesorado Universitario (FPU)

C. Martín-Vide et al. (Eds.): WMC 2003, LNCS 2933, pp. 1–18, 2004.

We recall that in the evolution–communication model the computation consists of two actions: the *evolution* of the symbol-objects (application of simple rewriting rules) and the *communication* between the regions (application of symport/antiport rules).

It has been shown in [2] that evolution–communication P systems are universal, using two membranes, non-cooperative evolution rules and symport and antiport rules of weight one.

The proton pumping model is obtained adding a biological restriction to the evolution-communication model and in particular over the *antiport rules*. Considering that, in many bacteria, the only antiports available are those that can exchange a proton with some chemical objects, a natural step is to add such restrictions to the model. Therefore, a proton pumping P system is an evolution–communication P system with a set of special objects called *protons* that are never created and never destroyed and where only antiports that can exchange some symbol-objects (also protons among them) for a single proton are admitted (inspired by biology, we call such antiport rules *proton pumping rules*).

The similarity between the *catalyst* objects and the protons should be noticed. In both cases such special objects are never created and never destroyed; while catalysts are used to help some evolution rule to be applied, the protons are used to help some communication rules to be applied.

We show that the proton pumping model is universal, providing that it can use *sufficient* types of *different* protons during the computation. The proof is made, again, using the simulation of programmed grammars with appearance checking as in [2].

Moreover we show that, when one can use only one or two types of different protons (and this is the case in biology) then the proton pumping model is at least as powerful as ET0L system, but using (total) *weak* or *strong* priority of antiport rules over symport and evolution rules.

Finally, inspired from the fact that in biology the possible different types of antiports and symports used are limited (actually there is a constant number of them), we have studied some *descriptional complexity* measures of the systems considered.

## 2 Definitions

We start by recalling from [3] the definition of an EC P system. It is given by the alphabet, the membrane structure, the multisets of symbol-objects in each region, the evolution rules and symport/antiport rules as formalized below.

**Definition 1.** *An* evolution–communication P system *(in short,* an EC P system*), of degree* $m \geq 1$*, is defined as*

$$\Pi = (O, \mu, w_0, w_1, w_2, \cdots, w_m, R_1, \cdots, R_m, \bar{R}_1, \cdots, \bar{R}_m, i_o),$$

*where:*

- *$O$ is the alphabet of objects;*
- *$\mu$ is a membrane structure with $m$ membranes injectively labeled with $1, 2, \cdots, m$;*
- *$w_i$ are strings which represent multisets over $O$ associated with regions $1, 2, \cdots, m$ of $\mu$ ($w_0$ represents the environment);*
- *$R_i$, $1 \leq i \leq m$, are finite sets of simple evolution rules over $O$; $R_i$ is associated with the region $i$ of $\mu$; a simple evolution rule is of the form $u \to v$, where $u$ and $v$ are strings over the alphabet $O$;*
- *$\bar{R}_i$, $1 \leq i \leq m$, are finite sets of symport/antiport rules over $O$; $\bar{R}_i$ is associated with the membrane $i$ of $\mu$;*
- *$i_o \in \{0, 1, 2, \cdots, m\}$ is the output region; if $i_o = 0$, then it is the environment, otherwise $i_o$ is a label of some membrane of $\mu$.*

The basic model assumes that all rules are applied in a nondeterministic, maximally parallel way and that there is no priority among the evolution and communication rules. The evolutive approach and the communicative approach were also proposed, as having (strong) priority of evolution rules and of communicative rules, respectively.

The following notation is used

$$NECP_m(i, j, \alpha), \alpha \in \{ncoo, coo\} \cup \{cat_k \mid k \geq 0\}$$
$$(PsECP_m(i, j, \alpha), \alpha \in \{ncoo, coo\} \cup \{cat_k \mid k \geq 0\})$$

to denote the family of sets of natural numbers (the family of sets of vectors of natural numbers) generated by EC P systems with at most $m$ membranes (as usually, $m = *$ if such a number is unbounded), using symport rules of weight at most $i$, antiport rules of weight at most $j$, and evolution rules that can be cooperative ($coo$), non-cooperative ($ncoo$), or catalytic ($cat_k$), using at most $k$ catalysts.

Now we are ready to give the definition of a proton pumping P system.

**Definition 2.** *A proton pumping P system of degree $m \geq 1$ is defined as*

$$\Pi = (O, P, \mu, w_0, w_1, \cdots, w_m, R_1, \cdots, R_m, R'_1, \cdots, R'_m, R''_1, \cdots, R''_m, i_o), \quad (1)$$

*where $(O, \mu, w_0, w_1, \cdots, w_m, R_1, \cdots, R_m, R'_1 \cup R''_1 = \bar{R}_1, \cdots, R'_m \cup R''_m = \bar{R}_m, i_o)$ is an evolution–communication P system, $P \subseteq O$ is the set of protons, $R'_i$ are the sets of symport rules and $R''_i$ are the sets of antiport rules (proton pumping rules) of the form $(x, out; p, in)$ or $(p, out; x, in)$ where $x \in O^+$ and $p \in P$. Every evolution rule is of the form $u \to v$, where $u \in (O - P)^+, v \in (O - P)^*$.*

The computation of a proton pumping P system evolves like in the case of an evolution–communication P system. The $m$-tuple of multisets of objects present at any moment in the regions of $\Pi$ represents the configuration of the system at that moment (the $m$-tuple $(w_1, \cdots, w_m)$ is the initial configuration). A transition between configurations is governed by the mixed application of the evolution rules and of the symport/antiport rules. All objects which can be the "subject"

of the rules from the sets $R_i, R'_j, R''_j$, $1 \leq i \leq m, 1 \leq j \leq m$, have to evolve by such rules. As usual, the rules from $R_i$ are applied to objects in region $i$ and the rules from $R'_i$ and $R''_i$ govern the communication of objects through membrane $i$. There is no difference between evolution rules and communication rules (symports and proton pumping rules): they are chosen and applied in the non-deterministic maximally parallel manner. The system continues parallel steps until there remain no applicable rules (evolution rules or symport/antiport rules) in any region of $\Pi$. Then the system halts, and we consider the multiplicities of objects contained in the output region $i_o$, at the moment when the system halts, as the result of the computation of $\Pi$. The set of all vectors of natural numbers computed in this way is denoted by $Ps(\Pi)$. Later we consider other ways to define a computation where we introduce some kind of priority among the rules involved.

We use the following notations

$$PsProP_m^k(i,j,\alpha), \alpha \in \{ncoo, coo\} \cup \{cat_k \mid k \geq 0\},$$

to denote the family of sets of vectors of natural numbers) generated by a proton pumping P systems with at most $m$ membranes (as usually, $m = *$ if such a number is unbounded), $k$ different types of protons (i.e., $k$ is the cardinality of the set $P$), using symport rules of weight at most $i$, antiport rules of weight at most $j$, and evolution rules that can be cooperative ($coo$), non-cooperative ($ncoo$), or catalytic ($cat_k$), using at most $k$ catalysts.

## 3 Variants: Weak/Strong Priority of (Proton) Pumping

After we have introduced the basic model of proton pumping P systems, we define here two basic variants, derived, to some extent, from biology, and give the notions of weak and strong priority as presented in [5].

The first variant is with *weak priority of proton pumping*, where a weak priority of proton pumping rules over other kinds of rules is assumed. In this case, weak priority means that in the process of assigning rules to objects, first the proton pumping rules are assigned in a nondeterministic, maximally parallel manner, and then the other rules are assigned to the remaining objects, again nondeterministically and in the maximally parallel way.

This is different from the strong priorities, as usually considered in P systems, because the rules with a lower priority, can be also applied in the same step as the proton pumping rules, if there are enough objects for them.

This differs also from the usually considered P systems with priorities, because here the priorities are *total*: they are specified as priorities of (all) proton pumping rules over (all) rewriting and symport rules, rather than between individual rules.

The second variant proposed is with *strong priority of pumping*. In this case the *total* strong priority is introduced in the following sense: (all) antiport rules associated to a membrane have strong priorities over (all) rewriting rules, associated to both regions adjacent to the membrane, and over (all) symport rules, moving objects from either region adjacent to the membrane.

In other words this means that if a pumping rule is applied in some membrane then it *blocks* all the other evolution and symport rules that could take objects from the two regions where the proton pumping rule chooses its objects.

In case of the weak priority and strong priority variants we use the notation:

$$PsProP_m^k(i,j,\alpha,wpp),\ \alpha \in \{ncoo, coo\} \cup \{cat_k \mid k \geq 0\},$$
$$[PsProP_m^k(i,j,\alpha,spp),\ \alpha \in \{ncoo, coo\} \cup \{cat_k \mid k \geq 0\}]$$

to denote the family of sets of vectors of natural numbers generated by proton pumping P systems with at most $m$ membranes ($m = *$ if such a number is unbounded), at most $k$ different types of protons, symport rules of weight at most $i$, antiport rules of weight at most $j$, and evolution rules that can be cooperative ($coo$), non-cooperative ($ncoo$), or catalytic ($cat_k$) with at most $k$ catalysts and weak [strong] pumping priority.

In what follows we only take in considerations proton pumping P systems with non-cooperative evolution rules, in the basic model and in the weak and strong priority variants. Moreover, it is known that, in reality, the possible number of different types of different antiport, symport and evolution rules used by some biological cell is limited by a constant (a fixed number of transport mechanisms is, for example, available in many bacteria); for this reason we think that it is useful to observe some of the *biological descriptional complexity* parameters of the systems considered. In particular, we will be interested in the total number of antiport, symport and evolution rules used by a proton pumping system. A detailed survey of these types of results obtained in this paper can be found in the Appendix.

Now we need to recall some preliminaries concepts and some notations. First, we introduce a useful normal form for ET0L systems.

**Lemma 1.** *For each $L \in ET0L$ there is an extended tabled Lindenmayer system $G = (V, T, H, w_0)$ with 2 tables ($H = \{h_1, h_2\}$) generating $L$, such that the terminals are only trivially rewritten: for each $a \in T$ if $(a \to \alpha) \in h_1 \cup h_2$, then $\alpha = a$.*

*Proof.* Let $L \in ET0L$. Then, there exists an ET0L system $G_0$ such that $L(G_0) = L$. For $G_0$, an equivalent ET0L system $G_1$ can be constructed, where all terminals are only trivially rewritten. For $G_1$, an equivalent ET0L system $G$, containing only two tables, can be constructed. Moreover, the transformation can be performed only on the nonterminals, leaving terminals as they are. □

We also give a slightly modified definition of programmed grammars with appearance checking, similar to the one used in [4]:

**Definition 3.** *A* programmed grammar *(with appearance checking) is a system $G = (N, T, S, P)$, where $N$ is a finite set of nonterminal symbols, $T$ is a finite set of terminal symbols ($N \cap T = \emptyset$), $S \in N$ is the start symbol and $P$ is a finite set of tuples of the form $(r : \alpha \to \beta, \sigma(r), \varphi(r))$, where $r$ is a label of a rewriting rule $\alpha \to \beta$,*

$$Lab(P) = \{r \mid (r : \alpha \to \beta, \sigma(r), \varphi(r))\}$$

*is the set of labels of rules in $P$, and $\sigma, \varphi : Lab(P) \longrightarrow 2^{Lab(P)}$; $\sigma(r), \varphi(r)$ are called the success field and the failure field of $r$, respectively.*

**Definition 4.** *The* language *generated by a programmed grammar.*
*Let $(r : \alpha \to \beta, \sigma(r), \varphi(r)) \in P$. We say that $w'$ is derived from $w$ in one step by applying or skipping the rule with label $r$ ($w \Rightarrow_r w'$) if either $w = x\alpha y$, $w' = x\beta y$ or $w = w'$, $\alpha \notin Sub(w)$. In the derivation, pairs of label and word are considered: $(r, w) \Rightarrow (r', w')$ if $w \Rightarrow_r w'$ and either $\alpha \in Sub(w)$ and $r' \in \sigma(r)$, or $\alpha \notin Sub(w)$ and $r' \in \varphi(r)$. In other words, if $\alpha$ is present in the sentential form, then the rule is used and the next rule to be applied is chosen from those with label in $\sigma(r)$, otherwise, the sentential form remains unchanged and we choose the next rule from the rules labeled by some element of $\varphi(r)$. Let $\Rightarrow^*$ be a reflexive and transitive closure of $\Rightarrow$. The language generated by a programmed grammar $G$ is $L(G) = \{x \in T^* \mid (r, S) \Rightarrow^* (r', w'), w' \Rightarrow_{r'} x\}$.*

*Remark 1.* In this definition it is natural to have $w' \Rightarrow_{r'} x$ rather than $(r', w') \Rightarrow (r'', x)$ because we need not to have the next rule after we have obtained the terminal string. If $w' = uAv$, $u, v \in T^*$, $A \in N$, $(r' : A \to y, \sigma(r'), \varphi(r')) \in P$, and $(r, S) \Rightarrow^* (r', w')$, then we say that $x = uyv$ belongs to the language $L(G)$, even if $\sigma(r') = \emptyset$. This definition is family-equivalent to the one with $(r', w') \Rightarrow (r'', x)$ because for any such grammar we could add a dummy rule $(r : S \to S, \emptyset, \emptyset)$ to $P$, and add $r$ to the success and failure fields of all terminal rules without changing the language. We take advantage of this fact in the universality proof.

If $\varphi(r) = \emptyset$ for each $r \in Lab(P)$, then the grammar is said to be *without appearance checking*. If $\sigma(r) = \varphi(r)$ for each $r \in Lab(P)$, then the grammar is said to be with *unconditional transfer*: in such grammars the next rule is chosen irrespective of whether the current rule can be effectively used or not).

*Note 1.* From now on by programmed grammars we will assume programmed grammars with appearance checking with *context-free rules*.

*Remark 2.* In the universality proof the programmed grammars with appearance checking will be simulated, considering pairs

$$\langle S = w_1, p_0 \rangle, \cdots, \langle w_m, p_{m-1} \rangle, \quad \langle w_{m+1} = x, p_m \rangle \quad \text{for}$$
$$(p_1, S = w_1) \Rightarrow \cdots \Rightarrow (p_m, w_m),\ w_m \Rightarrow_{p_m} w_{m+1} = x\ .$$

Here, the rule is chosen during the step when it is applied/skipped, rather than one step before. $p_0$ is a new symbol - a starting point of the control sequence.

During the proof of all the theorems presented in this paper, we will indicate with the label $(nx)$ the rule (evolution or antiport/symport rule) present in equation $n$ and corresponding to the alphabetical place $x$ (for example: $(12b)$ means "the second rule in equation (12)").

## 4 Universality of Proton Pumping P Systems

In this section we give a universality theorem for proton pumping P systems. We prove that such systems are universal when they can use *sufficient* types of protons during the computation. The proof is based on the simulation of programmed grammars with appearance checking. We recall the following lemma (Theorem 1.2.5, [4]):

**Lemma 2.** *The programmed grammars with appearance checking generate exactly the family of recursively enumerable languages.*

Finally we recall that the notation $Perm(x)$ indicates the set of all strings that can be obtained as a permutation of the string $x$.

**Theorem 1.** $PsProP_3^*(1, \mathbb{1}, ncoo) = PsRE$.

*Proof.* Let $L \in RE$. Then there exists a programmed grammar with appearance checking $G = (N, T, P, S)$ generating $L$, where $Lab(P) = \{i \mid 1 \leq i \leq m\}$ and $N = \{X_i \mid 1 \leq i \leq n\}$. We will need the following notations: $N' = N \cup \{h\}$, $\overline{N'} = \{\overline{X} \mid X \in X'\}$. We now define the morphism $\gamma : (N' \cup T)^* \longrightarrow (\overline{N'} \cup T)^*$ by $\gamma(x) = \overline{x}, x \in N'$, and $\gamma(x) = x$, $x \in T$. Let us construct the P system

$$
\begin{aligned}
\Pi &= (O, P, \mu, \varepsilon, w_1, w_2, w_3, R_1, R_2, R_3, R_1', R_2', R_3', \emptyset, R_2'', R_3'', 0),\\
P &= \{p_i, q_i \mid i \in Lab(P)\} \cup \{r_j \mid X_j \in N\} \cup \{p_0, c, F\},\\
\tilde{N'} &= \{\tilde{X} \mid X \in N'\},\\
C &= \{b_i^{(j)}, d_i^{(j)} \mid i \in Lab(P), 1 \leq j \leq 5\} \cup \{l, g, H\},\\
M &= \{t_j \mid -2 \leq j \leq 5\} \cup \{\#, K, s\},\\
A &= \{e_j, e_j' \mid X_j \in N\} \cup \{f_j \mid 1 \leq j \leq 5\},\\
O &= P \cup T \cup N' \cup \tilde{N'} \cup \overline{N'} \cup C \cup A \cup M,\\
\mu &= [_1[_2[_3]_3]_2]_1,\\
w_1 &= Fsr_1r_2 \cdots r_n,\\
w_2 &= p_0cK,\\
w_3 &= Shp_1p_2 \cdots p_mq_1q_2 \cdots q_m,\\
R_1 &= \{\tilde{X} \to \gamma(x)d_i^{(5)} \mid (i : (X \to x, \sigma(i), \varphi(i))) \in P\}\\
&\quad \cup \{\tilde{h} \to \overline{h}b_i^{(5)}e_jf_5 \mid (i : (X_j \to x, \sigma(i), \varphi(i))) \in P\}\\
&\quad \cup \{\tilde{X} \to \# \mid X \in N\} \cup \{f_j \to f_{j-1} \mid 2 \leq j \leq 5\}\\
&\quad \cup \{b_i^{(5)} \to \#, d_i^{(5)} \to \# \mid i \in Lab(P)\} \cup \{\# \to \#\}, \qquad (2)\\
R_2 &= \{X \to \tilde{X}t_5 \mid X \in N'\} \cup \{t_j \to t_{j-1} \mid -1 \leq j \leq 5\}\\
&\quad \cup \{b_i^{(5)} \to b_i^{(4)}lg, d_i^{(5)} \to d_i^{(4)}lg \mid i \in Lab(P)\}\\
&\quad \cup \{b_i^{(j)} \to b_i^{(j-1)}, d_i^{(j)} \to d_i^{(j-1)} \mid i \in Lab(P), 2 \leq j \leq 4\}\\
&\quad \cup \{e_j \to e_j' \mid X_j \in N\}, \qquad (3)
\end{aligned}
$$

$$R_3 = \{\overline{X} \to X \mid X \in N'\} \cup \{g \to H, K \to K\}, \tag{4}$$
$$R'_1 = \{(a, out) \mid a \in T\}, \tag{5}$$
$$R'_2 = \{(\tilde{X}, out), (\overline{X}, in) \mid X \in N'\} \cup \{(e_j, in) \mid X_j \in N\}, \tag{6}$$
$$R'_3 = \{(\overline{X}, in) \mid X \in N'\} \cup \{(g, in)\}, \tag{7}$$
$$\begin{aligned} R''_2 = {} & \{(p_j, out; d_i^{(5)}, in), (p_j, out; b_i^{(5)}, in) \mid i \in \sigma(j)\} \\ & \cup \{(q_j, out; d_i^{(5)}, in), (q_j, out; b_i^{(5)}, in) \mid i \in \varphi(j)\} \\ & \cup \{(p_0, out; d_i^{(5)}, in), (p_0, out; b_i^{(5)}, in) \mid i \in Lab(P)\} \\ & \cup \{(l, out; p_j, in) \mid j \in Lab(P) \cup \{0\}\} \\ & \cup \{(l, out; q_j, in) \mid j \in Lab(P)\} \\ & \cup \{(e'_j, out; r_j, in), (r_j, out; f_1, in) \mid X_i \in N\} \\ & \cup \{(c, out; s, in), (s, out; F, in)\}, \end{aligned} \tag{8}$$
$$\begin{aligned} R''_3 = {} & \{(X, out; c, in) \mid X \in N'\}, \\ & \cup \{(p_i, out; d_i^{(1)}, in), (q_i, out; b_i^{(1)}, in) \mid i \in Lab(P)\} \\ & \cup \{(H, out; p_i, in) \mid i \in Lab(P) \cup \{0\}\} \\ & \cup \{(H, out; q_i, in) \mid i \in Lab(P)\} \\ & \cup \{(X_j, out; r_j, in), (r_j, out; K, in) \mid X_j \in N\} \\ & \cup \{(c, out; t_{-2}, in), (X, out; F, in), (F, out; K, in)\}. \end{aligned} \tag{9}$$

The system $\Pi$ simulates the programmed grammar $G$ (see also [2]) using the following idea. The protons $(p_i, q_i)$ serve the role of remembering the label of the rule that has been previously applied $(p_i)$ or skipped $(q_i)$, appearance checking $(r_j)$, $p_0$ is the starting point of the control sequence, $c$ is used to sequentialize the application of the rules, and $F$ checks that all nonterminals were rewritten at the end.

We also use objects associated with each terminal $(T)$, associated with each non-terminal $(N')$, including also the rule application failure symbol $(h)$, their "first" versions $(\overline{N'})$, their intermediate versions $(\tilde{N'})$, control sequence symbols ($b_i^{(j)}$ if the rule with label $j$ was skipped and $d_i^{(j)}$ if the rule with label $j$ was applied), appearance checking symbols $(e_j, e'_j)$. Objects $f_j$ are used to return the appearance checking protons, $l, g, H$ are used to return the control sequence proton, $t_j$ are delaying $c$ for synchronization, $K$ helps $F$ and $r_i$ to block the computation, $s$ is used to end the simulation of derivation, switching to checking that no more nonterminals are in region 3, and # is the trap symbol.

In region 1 (the outer one) we simulate the application of the context-free rules of $G$, using simple evolution rules (2a), while the rules (8a-8f) enforce the "control sequence", i.e., transitions from an application of (or from the failure to apply) a rule of $G$ to the next one. During the simulation, the nonterminals to be rewritten are brought into region 1 by rules (9a, 3a, 6a) and the result is returned by rules (6b, 7a), except the terminal symbols, ejected into the environment by (5). Rules (4a) remove the bars. The failure to apply a rule is simulated by (2b). The appearance checking is enforced by the rules (6c, 3g, 8i, 9f). If